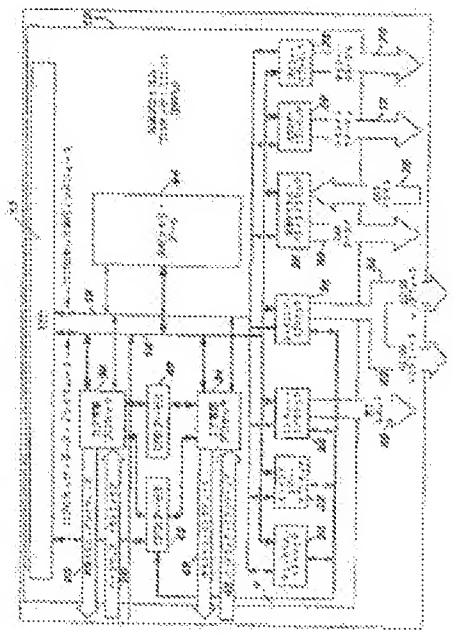
A PLURALITY OF LOGICAL INTERFACES TO SHARED COPROCESSOR RESOURCE

Publication number: JP2002149424 (A)
Publication date: 2002-05-24
Inventor(s): DAVIS GORDON TAYLOR; MARKO C HEADS; RIWENS ROSE BOYED; RINALDI MARK ANTHONY
Applicant(s): IBM
Classification:
- international: *G06F9/38; G06F9/46; G06F13/10; G06F13/14; G06F15/00; G06F15/16; G06F15/163; G06F15/76; G06F15/78; G06F15/80; H04L29/06;* G06F9/38; G06F9/46; G06F13/10; G06F13/14; G06F15/00; G06F15/16; G06F15/76; H04L29/06; (IPC1-7): G06F9/46; G06F9/38
- European: G06F15/78P2
Application number: JP20010265792 20010903
Priority number(s): US20000656582 20000906

Also published as:
US6829697 (B1)
TW581950 (B)
SG100751 (A1)
KR20020020186 (A)
CN1342940 (A)
more >>

Abstract of **JP 2002149424 (A)**

PROBLEM TO BE SOLVED: To increase the communicating efficiency of a protocol processor unit(PPU) and a coprocessor in a network processor. SOLUTION: An integrated processor composite body is provided with a plurality of protocol processor units(PPU). The respective units are provided with at least one or preferentially two individually functioning core language processors (CLP). The respective CLP are allowed to support dual threads through a logical coprocessor execution/data interface with a plurality of exclusive coprocessors to be used for the respective PPU. In response to an operation instruction, the PPU identifies an events whose waiting time is long and an event whose waiting time is short, and controls and switches the priority order of the execution of threads based on the identification. Also, in response to the operation instruction, the conditional execution of the specific coprocessor operation is made available when the designated specific event is generated or not generated.



Data supplied from the **esp@cenet** database — Worldwide

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号
特開2002－149424
（P2002－149424A）
(43)公開日　平成14年５月24日(2002.5.24)

(51)Int.Cl.'		識別記号	FI		テーマコード'(参考)
Ｇ０６Ｆ	9/46	３６０	Ｃ０６Ｆ	9/46	３６０Ｂ　５Ｂ０１３
					３６０Ｃ　５Ｂ０９８
	9/38	３７０		9/38	３７０Ｃ

審査請求　有　　請求項の数29　ＯＬ　（全 21 頁）

(21)出願番号　特願2001－265792(P2001－265792)

(22)出願日　平成13年９月３日(2001.9.3)

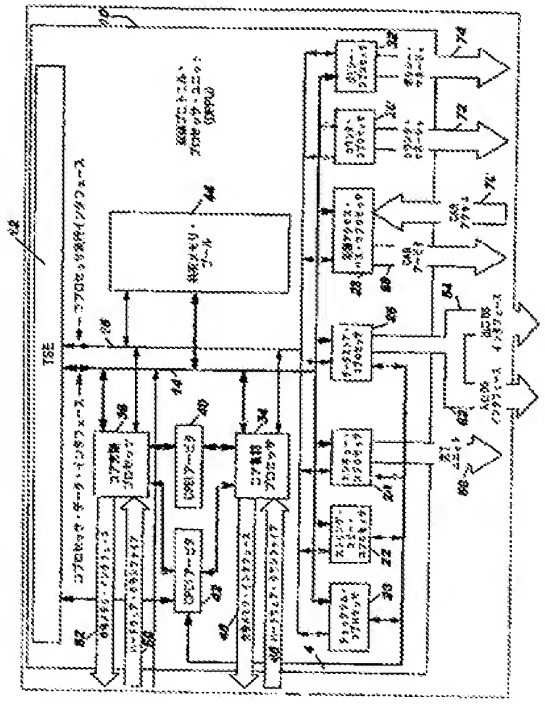
(31)優先権主張番号　０９／６５６５８２
(32)優先日　平成12年９月６日(2000.9.6)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）
(74)代理人　100086243
弁理士　坂口　博　（外２名）

最終頁に続く

(54)【発明の名称】　共有コプロセッサ・リソースに対する複数の論理インタフェース

(57)【要約】　（修正有）
【課題】ネットワーク・プロセッサにおいて、プロトコル・プロセッサ・ユニット（ＰＰＵ）とコプロセッサとの通信効率を上げる。
【解決手段】　組み込みプロセッサ複合体は複数のプロトコル・プロセッサ・ユニット（ＰＰＵ）を含む。各ユニットに少なくとも１つの、好適には２つの個別に機能するコア言語プロセッサ（ＣＬＰ）が含まれる。各ＣＬＰは、各ＰＰＵに用いられる複数の専用コプロセッサとの論理コプロセッサ実行／データ・インタフェースを通してデュアル・スレッドをサポートする。操作命令により、ＰＰＵが待ち時間の長いイベントと短いイベントを識別し、この識別をもとにスレッド実行の優先順位を制御し切り替える。また操作命令により、指定された特定のイベントの発生時または非発生時、特定のコプロセッサ・オペレーションの条件付き実行が可能になる。



【特許請求の範囲】
【請求項１】ネットワーク・プロセッサのプログラミング機能を制御する組み込みプロセッサ複合体のオペレーションであって、該プロセッサ複合体は、複数のプロトコル・プロセッサ・ユニット（ＰＰＵ）を含み、各ＰＰＵは少なくとも１つのコア言語プロセッサ（ＣＬＰ）を含み、各ＣＬＰは少なくとも２つのコード・スレッドを持ち、各ＰＰＵはＰＰＵの特定のタスクを実行する上で有用な複数のコプロセッサ及び複数の論理コプロセッサ・インタフェースを利用し、各ＣＬＰと該コプロセッサ間のアクセスを実現する、オペレーション。
【請求項２】前記コプロセッサは、各ＣＬＰの複数のコード・スレッドをサポートする専用コプロセッサを含む、請求項１記載のオペレーション。
【請求項３】前記コプロセッサは、ツリー検索コプロセッサ、チェックサム・コプロセッサ、ストリングコピー・コプロセッサ、エンキュー・コプロセッサ、データストア・コプロセッサ、ＣＡＢコプロセッサ、カウンタ・コプロセッサ、及びポリシ・コプロセッサを含むグループから選択される、請求項１記載のオペレーション。
【請求項４】複数のスレッド間の優先順位を確認するためコプロセッサ実行インタフェース・アービタを含む、請求項３記載のオペレーション。
【請求項５】データ・スレッド間の優先順位を確認するためコプロセッサ・データ・インタフェース・アービタを含む、請求項３記載のオペレーション。
【請求項６】各スレッドと少なくとも１つのコプロセッサ間にＦＩＦＯバッファを含む、請求項３記載のオペレーション。
【請求項７】前記ＦＩＦＯバッファは各スレッドと前記カウンタ・コプロセッサの間にある、請求項６記載のオペレーション。
【請求項８】前記ＦＩＦＯバッファは各スレッドと前記ポリシ・コプロセッサの間にある、請求項６記載のオペレーション。
【請求項９】ネットワーク・プロセッサのプログラミング機能を制御する組み込みプロセッサ複合体を含むネットワーク処理システムであって、該複合体は複数のプロトコル・プロセッサ・ユニット（ＰＰＵ）を含み、各ＰＰＵは、
それぞれ少なくとも２つのコード・スレッドを持つ少なくとも１つのコア言語プロセッサ（ＣＬＰ）と、
前記システムの特定のタスクを実行する複数のコプロセッサ及び該コプロセッサのリソースにアクセスし各ＣＬＰと共有する複数のコプロセッサ・インタフェースと、
を含む、システム。
【請求項１０】前記コプロセッサ・インタフェースは、各ＣＬＰのコード・スレッドをサポートすることにのみ用いられる、請求項９記載のネットワーク処理システム。
【請求項１１】前記コプロセッサは、ツリー検索コプロセッサ、チェックサム・コプロセッサ、ストリングコピー・コプロセッサ、エンキュー・コプロセッサ、データストア・コプロセッサ、ＣＡＢコプロセッサ、カウンタ・コプロセッサ、及びポリシ・コプロセッサを含むグループから選択される、請求項１０記載のネットワーク処理システム。
【請求項１２】各スレッドと前記コプロセッサのうち少なくとも１つの間にＦＩＦＯバッファを含む、請求項１０記載のネットワーク処理システム。
【請求項１３】前記ＦＩＦＯバッファは各スレッドと前記カウンタ・コプロセッサの間にある、請求項１２記載のネットワーク処理システム。
【請求項１４】前記ＦＩＦＯバッファは各スレッドと前記ポリシ・コプロセッサの間にある、請求項１２記載のネットワーク処理システム。
【請求項１５】前記ＣＬＰのスレッドにより実行される特定の操作命令を含み、該実行の結果、コプロセッサ・オペレーションを制御するコマンドが得られ、該コマンドは前記ＣＬＰとコプロセッサ間のインタフェースを流れる、請求項９記載のネットワーク処理システム。
【請求項１６】前記命令は、特定のコプロセッサ・オペレーションの条件付き実行を可能にするように働く、請求項１５記載のネットワーク処理システム。
【請求項１７】前記命令により、前記システムが、特定のコプロセッサ・コマンドに応答してデータにアクセスするための予測応答時間に従って、待ち時間の長いイベントと待ち時間の短いイベントを識別し、アクティブなスレッドの実行が待ち時間の長いイベントにより中断したときに他のスレッドにフル制御を与えるか、またはアクティブなスレッドの実行が待ち時間の短いイベントにより中断したときに他のスレッドに一時的制御を与える、請求項１５記載のネットワーク処理システム。
【請求項１８】複数のプロトコル・プロセッサ・ユニット（ＰＰＵ）を含む組み込みプロセッサ複合体内の命令の実行を制御する方法であって、該プロトコル・プロセッサ・ユニットはそれぞれ少なくとも１つのコア言語プロセッサ（ＣＬＰ）を含み、該ＣＬＰはそれぞれ少なくとも２つのコード・スレッドを持ち、該方法は、該ＰＰＵに対する特定のタスクを実行するため、各ＰＰＵによる複数のコプロセッサの使用、及び該コプロセッサと各ＣＬＰ間のアクセスを提供する複数の論理コプロセッサ・インタフェースの使用を含む、方法。
【請求項１９】前記ＰＰＵの複数のコード・スレッドをサポートする専用コプロセッサの使用を含む、請求項１８記載の方法。
【請求項２０】前記コプロセッサの1つ以上は、ツリー検索コプロセッサ、チェックサム・コプロセッサ、ストリングコピー・コプロセッサ、エンキュー・コプロセッサ、データストア・コプロセッサ、ＣＡＢコプロセッ

サ、カウンタ・コプロセッサ、及びポリシ・コプロセッサを含むグループから選択される、請求項１９記載の方法。

【請求項２１】実行スレッド間の優先順位を確認するためコプロセッサ実行インタフェース・アービタが用いられる、請求項２０記載の方法。

【請求項２２】データ・スレッド間の優先順位を確認するためコプロセッサ・データ・インタフェース・アービタが用いられる、請求項２０記載の方法。

【請求項２３】各スレッドと前記コプロセッサのうち少なくとも１つの間にＦＩＦＯバッファを提供するステップを含む、請求項２０記載の方法。

【請求項２４】前記ＦＩＦＯバッファは各スレッドと前記カウンタ・コプロセッサの間にある、請求項２３記載の方法。

【請求項２５】前記ＦＩＦＯバッファは各スレッドと前記ポリシ・コプロセッサの間にある、請求項２３記載の方法。

【請求項２６】前記ＣＬＰにより実行される特定の操作命令を提供するステップを含み、該実行の結果、コプロセッサ・オペレーションを制御するコマンドが得られ、該コマンドは前記ＣＬＰとコプロセッサの間のインタフェースを流れる、請求項１８記載の方法。

【請求項２７】前記操作命令により特定のコプロセッサ・オペレーションの条件付き実行が可能になる、請求項２６記載の方法。

【請求項２８】前記実行は直接的または間接的である、請求項２７記載の方法。

【請求項２９】前記システムが、特定のコプロセッサ・コマンドに関する予測応答時間に従って、待ち時間の長いイベントと待ち時間の短いイベントを識別し、アクティブなスレッドの実行が待ち時間の長いイベントにより中断したときに他のスレッドにフル制御を与えるか、またはアクティブなスレッドの実行が待ち時間の短いイベントにより中断したときに他のスレッドに一時的制御を与える、命令を提供するステップを含む、請求項１８記載の方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、ネットワーク・プロセッサ・システムに関し、特に１つ以上のプロトコル・プロセッサ・ユニット（ＰＰＵ）を含む組み込みプロセッサ複合体に関する。プロセッサ装置内でＰＰＵと複数のコプロセッサを相互接続するインタフェースを通して、それらの間でデータや命令を転送するため、ＰＰＵとともに複数のコプロセッサが用いられる。

【０００２】

【従来の技術】ネットワーク・プロセッサのプログラミング機能を実現し、これを制御するためプロトコル・プロセッサ・ユニットが用いられていることは周知の通りである。同様に、コンピュータ・システム処理複合体アーキテクチャを設計する際に、ＰＰＵとともにコプロセッサが用いられることも一般的になっている。リアルタイム処理を必要とする処理イベントの遅れは、システム性能に直接影響を与える問題である。ＰＰＵによりタスクを実行するのではなく、特定のコプロセッサにタスクを割当てることによって、コンピュータ・システムの効率と性能を上げることができる。ＰＰＵがコプロセッサと効率よく通信することが重要である。この通信を改良し続けることが求められる。

【０００３】

【発明が解決しようとする課題】本発明の目的は、１つ以上のコア言語プロセッサ（ＣＬＰ）を含み、各ＣＬＰが複数のスレッドを持ち、論理コプロセッサ・インタフェースを通して特別タスク・コプロセッサに指示を与えるプロトコル・プロセッサ・ユニット（ＰＰＵ）を使用することである。

【０００４】本発明の他の目的は、共有コプロセッサ・リソースにアクセスするため（プログラマから見て）複数の論理コプロセッサ・インタフェースを使用することである。コプロセッサ・リソースは、ＰＰＵ内の複数の処理スレッドにより共有されることがあり、複数のＰＰＵ間で１つのコプロセッサ・リソースが共有されることもある。

【０００５】本発明の他の目的は、ＰＰＵとそのコプロセッサのインタフェース側で有効になる特定の操作に関する。この操作の１つは、コプロセッサ命令を条件付きで実行する機能である。これは特にカウンタ・コプロセッサで有効であるが、一般には他のコプロセッサにも適用できる。コプロセッサ・インタフェースは、特定のコプロセッサ・コマンドに関する予測応答時間に従って、待ち時間の長いイベントと短いイベントを識別する機能を持つ。この識別により、スレッドを実行する優先順位が制御される。

【０００６】本発明の他の目的は、これまでのコプロセッサ・インタフェースに比べてフレキシビリティと効率が向上したコプロセッサ・インタフェースを提供することである。

【０００７】

【課題を解決するための手段】以上の目的は、他の目的を含めて、以下に述べるようにして達成される。

【０００８】ネットワーク・プロセッサのプログラミング機能を制御する組み込みプロセッサ複合体の動作について説明する。プロセッサ複合体は、複数のプロトコル・プロセッサ・ユニット（ＰＰＵ）を含み、ＰＰＵはそれぞれ１つ以上のコア言語プロセッサ（ＣＬＰ）を含む。ＣＬＰはそれぞれ複数のコード・スレッドを持つ。各ＰＰＵが、ＰＰＵに対して特定のタスクを実行する上で有用な複数のコプロセッサを利用する。複合体は、複数の論理コプロセッサ・インタフェースを使用し、ＣＬ

Pとの共有コプロセッサ・リソースにアクセスする。CLPにより特定の動作命令が実行され、これによりコプロセッサにコマンドが送られる。これらの命令の1態様は、特定のコプロセッサ命令の条件付き実行を可能にすることである。命令は、特定のコプロセッサ・コマンドに関する予測応答時間に従って、待ち時間の長いイベントと短いイベントを識別することができる。これにより複合体は、処理されている待ち時間の長さとタイプに応じて、制御をスレッドからスレッドへ切り替えることができる。

【0009】

【発明の実施の形態】本発明については、ネットワーク・プロセッサのプログラミング機能を提供し制御する埋め込みプロセッサ複合体の文脈で説明する。複合体の実施形態は通常、ハードウェア・アクセラレータと連携して高速パターン検索、データ操作、内部チップ管理機能、フレーム解析、及びデータのプリフェッチをサポートする8つのメイン処理ユニットまたはプロトコル・プロセッサ・ユニット（PPU）を含む。各PPUはそれぞれ構造コンポーネントを含み、構造コンポーネントは2つのCLPと、少なくとも1つ、好適には数個の専用／共有コプロセッサ・ユニット、及びメイン処理ユニットと各コプロセッサ・ユニットのインタフェースを含む。

【0010】各コプロセッサ・ユニットは、特定のネットワーク・タスクを実行することができる。メイン処理ユニットは、記憶プログラムの一連の命令を実行する。コプロセッサ・ユニットはそれぞれ、該メイン処理ユニットを担当し、メイン処理ユニットの制御下で特定のタスクを効率よく実行するようにされる。メイン処理ユニットと各コプロセッサ・ユニットのインタフェースにより、次の機能のうち1つ以上が有効になる。各コプロセッサ・ユニットの構成、各コプロセッサ・ユニットにより完了する特定のタスクの起動、各コプロセッサ・ユニットに関するステータス情報へのアクセス、及び各コプロセッサ・ユニットにより完了する特定のタスクに関する結果を返す手段の提供である。メイン処理ユニットとコプロセッサ・ユニットはそれぞれ1つ以上の専用レジスタを含む。インタフェースは、専用レジスタを該メイン処理ユニットとコプロセッサ・ユニットから共通アドレス・マップにマップすることができる。

【0011】各PPUはそれぞれ1つ以上のコア言語プロセッサ（CLP）及び数個の専用コプロセッサを含む。PPUに複数のCLPが含まれるとき、コプロセッサはCLP間で共有される。

【0012】各CLPはそれぞれ1つの演算論理ユニット（ALU）を含み、2つのコード・スレッドをサポートする（PPU毎に合計4つのスレッド）。CLPとコプロセッサは、スカラ・レジスタとアレイ・レジスタを含む専用レジスタの個別コピーを格納する。特定のコプロセッサは、CLPからコプロセッサのアレイ・レジスタやスカラ・レジスタへのコプロセッサ・コマンドの転送を調整するFIFOバッファを含む。CLPは一度に1つのスレッド・コマンドのみ実行する。プログラマから見ると、各スレッドはそれぞれ自体のコプロセッサまたはコプロセッサ・セットに見える。コプロセッサのほとんどは、専用機能を実行し、相互に並行して、またCLPとともに動作することができる。

【0013】PPUの制御記憶は、通常、内部メモリと外部メモリの両方により与えられる。例えば即時アクセスには内部SRAMを、高速アクセスには外部ZBT SRAMを、大容量が求められる場合にはDDR SDRAMを使用できる。

【0014】図1に、ツリー検索エンジン12、チェックサム・コプロセッサ20、ストリング・コピー・コプロセッサ22、エンキュー・コプロセッサ24、データストア・コプロセッサ26、制御アクセス・バス・コプロセッサ28、カウンタ・コプロセッサ30、及びポリシ・コプロセッサ32を含む数個のコプロセッサとのデータ・インタフェース14と実行インタフェース16を維持するプロトコル・プロセッサ・ユニット（PPU）10を示す。

【0015】PPUは、コア言語プロセッサ（CLP）のペア34及び36を含む。各CLPに命令フェッチ・デコード／実行ユニット、複数の専用レジスタ、汎用レジスタ、及び2つのスレッドが含まれる。コプロセッサ実行インタフェース（CPE1）アービタ40は、2つのCLPとコプロセッサ間で命令を調停する。コプロセッサ・データ・インタフェース（CPDI）アービタ42は、コプロセッサとCLP34及び36間で通信の優先順位を確認する。CLPの命令は全て命令メモリ（図1には示していない）に保存される。

【0016】CLP#1 34は、バス46を通してハードウェア・クラシファイア（classifier）からの通信を受信する。ハードウェア・クラシファイアは、ディスパッチャからの刺激を与え、新しいパケットの処理を開始する。CLPは、命令メモリからのフェッチ命令をバス48を通して送り、新しいパケットを処理する。同様にCLP36は、バス50を通してハードウェア・クラシファイアから分類結果を受信し、バス52を通して命令メモリ・インタフェースに要求を送る。命令メモリ・インタフェースとハードウェア・クラシファイアはPPUの外部にあり、図1には示していない。

【0017】エンキュー・コプロセッサ24はバス60を通して外部完了ユニット（図示せず）に命令を送る。データストア・コプロセッサ26は、データをバス62を通して入口データストア・インタフェースに、またはバス64を通して出口データストア・インタフェースに送る。制御データのフローは、バス68を通して外部制御アクセス・バス・アービタ（図示せず）により調停さ

れる。アクセスの調整は、書込まれているか読取られているデータがバス７０を通して流れている間にＣＡＢアービタ６８上で行われる。データ・アクセスはバス７０を通して入力または出力される。カウンタ・コプロセッサ３０とポリシ・コプロセッサ３２はそれぞれ、カウンタ・マネージャとポリシ・マネージャへのアクセスを、それぞれバス７２及び７４を通して提供する。

【００１８】図２に、本発明に従った２つのＣＬＰの他の詳細を示す。各ＣＬＰはそれぞれ汎用レジスタ８０と専用レジスタ８２を含む。これら専用レジスタは複数のスカラ・レジスタ８４とアレイ・レジスタ８６を含む。また命令フェッチ／デコード／実行の機能８８も含む。

【００１９】処理複合体は次のコンポーネントを含む。・複数のプロトコル・プロセッサ・ユニット。好適な実施例では、サーバが８つのプロトコル・プロセッサ・ユニット（ＰＰＵ）を使用する。各ＰＰＵに、複数（図では７つ）のコプロセッサを共有する１つ以上のＣＬＰが含まれる。ＰＰＵは、フレームを転送し、テーブルを更新し、ネットワーク処理ユニットを維持するためのコードを実行する。

【００２０】ＣＬＰは、共通命令メモリに保存されたコードを並行して実行する。各ＣＬＰはコアと３ステージのパイプライン、１６のＧＰＲ（汎用レジスタ）、及びＡＬＵ（演算論理ユニット）を含む。コプロセッサは操作を互いに並行して、またＣＬＰと並行して実行することができる。コプロセッサは、ＣＬＰとインタフェースを取る際、基本ＣＬＰ命令とレジスタ・モデルを拡張する。各コプロセッサのコマンドは新しいアセンブラ・ニーモニックとして現れ、コプロセッサのレジスタは、ＣＬＰプログラマから見て新しいスカラ・レジスタ、及びアレイ・レジスタとして現れる。アレイ・レジスタの一部は共有メモリ・プール４４に位置する。コプロセッサはＣＬＰと非同期に実行可能である。これによりＣＬＰは、コプロセッサがコマンドを実行している間に命令の処理を続けることができる。ＣＬＰは、待機命令によりコプロセッサのコマンド実行が完了するまで待機する。

【００２１】命令メモリ５６は通常、８つの組み込みＲＡＭで構成される。これらは、初期化時にロードされ、フレームを転送し、システムを管理するプログラム命令を格納する。命令メモリは、データ転送タスク、管理トラフィック、及び制御タスクのため１６Ｋの命令を保持する。

【００２２】ディスパッチャ・ユニット。これはスレッドの使用を管理し、新しいフレームをフェッチしアイドル・スレッドにディスパッチする。データ・フレームは、次に使用できるＰＰＵにディスパッチされる。これによりアップ／ダウン・ディスパッチャ・キュー（ｕｐ－ＧＤＱ、ｕｐ－ＧＣＱ、ｄｎ－ＧＲＯ／１、ｄｎ－ＧＢＯ／１、及びｄｎ－ＧＣＱ）からフレーム・アドレスのエンキューが解除される。エンキュー解除の後、ディスパッチャ・ユニットはアップ／ダウン・データストア（ＤＳ）からフレーム・ヘッダの一部を読取り、これを共有メモリ・プール４４に保存する。ＣＬＰがアイドルになるとすぐ、ディスパッチャ・ユニットがコード命令アドレス（ＣＩＡ）等の対応する制御情報をバス４６または５０を介してＣＬＰに渡す。ディスパッチャは別のバス５８を使用してヘッダ情報を共有メモリ・プール４４に送る。ディスパッチャはまたタイマと割込みを処理するため、それらの機能の作業を利用できるスレッドにディスパッチする。

【００２３】ツリー検索メモリ（ＴＳＭ）アービタ１１０。内部と外部に複数の共有メモリ位置があり、各ＣＬＰから利用できる。このメモリは共有されるので、アービタはメモリ・アクセス制御に用いられる。ＴＳＭはコードによって直接アクセスでき、例えばＴＳＭにルーティング・テーブルを保存するため使用できる。またＴＳＭは、ツリー検索時にＴＳＥ１２によりアクセスされる。

【００２４】完了ユニット（ＣＵ）。完了ユニットは２つの機能を実行する。第１に、ＣＬＰとＵｐ／Ｄｎ　ＥＤＳ（エンキュー、エンキュー解除、及びアイランドのスケジュール（Schedule Island）のインタフェースを取る。ＥＤＳはエンキュー操作を実行し、これによりフレーム・アドレスがＦＣＢＰａｇｅと呼ばれるパラメータとともに転送キュー、破棄キュー、またはディスパッチャ・キューにエンキューされる。ターゲットが転送キューのとき、ハードウェアにより構成されるフロー制御機構によって、フレームを転送キューにエンキューするか、破棄キューにエンキューするか確認される。第２に、完了ユニットはフレーム・シーケンスを保証する。同じフローに属するフレームは複数のスレッドによって処理される可能性があるので、それらのフレームがＵｐ／Ｄｎ転送キューに正しい順序でエンキューされるような予防措置が必要である。完了ユニットは、フレーム・ディスパッチ時にハードウェア・クラシファイア５４により生成されるラベルを使用する。

【００２５】ハードウェア・クラシファイア。これはディスパッチ・ユニットとＰＰＵの間のデータ・パスに置かれる。分類を行い、宛先スレッドに情報を提供する。Ｕｐフレームの場合、ハードウェア・クラシファイア・アシストがフレーム・フォーマットの周知のケースについて分類を行う。分類結果は、フレーム・ディスパッチの間、ＣＩＡ（コード命令アドレス）及び４つのＧＰＲ（汎用レジスタ）の内容の形でＣＬＰに渡される。Ｄｎフレームの場合、ハードウェア・クラシファイア・アシストが、フレーム・ヘッダに応じてＣＩＡを確認する。ハードウェア・クラシファイア・アシストは、Ｕｐ及びＤｎ両方のフレーム・ディスパッチでは、フレーム・シーケンスを維持するため完了ユニットにより用いられるラベルを生成する。

【００２６】Up／Dnデータストア・インタフェースとアービタ。各スレッドからデータストア・コプロセッサ２６を通してUp／Dnデータストアにアクセスすることができる。"他のデータ"を読取るときは読取りアクセスができ、データプールの内容をデータストアに書戻すときは書込みアクセスができる。複数のスレッドがあり、Upデークストアに一度にアクセスできるのは１スレッドのみ、Dnデータストアに一度にアクセスできるのも１スレッドのみなので、データストア毎に１つのアービタが必要である。

【００２７】制御アクセス・バス（CAB）アービタとWEBWatchインタフェース。CABアービタはCABへのアクセスをスレッド間で調停する。スレッドは全て、CABコプロセッサ２８を通してCABにアクセスすることができる。これによりプロセッサ・ユニットにある全てのメモリ及びレジスタの機能にアクセスできる。またどのスレッドも、全ての構成領域を変更または読取ることができる。CABは、プロセッサ・ユニットのメモリ・マップと考えることができる。

【００２８】CABWatchインタフェースは、３つのチップI／Oを使用して、チップ外部からCAB全体へのアクセスを提供する。

【００２９】デバッグ、割込み、シングル・ステップ制御、CABにより、GCHスレッドまたはCABWatchが各スレッドを制御することができる。例えばGPHスレッドまたはCABWatchがCABを使用して、シングル・ステップ実行モードで選択されたスレッドを実行することができる。

【００３０】コア言語プロセッサ（CLP）：ネットワーク・サーバは、そのアーキテクチャで数種類のCLPを利用する。CLPの種類は、それぞれ特定の機能を処理するようにプログラムされる。

【００３１】GDHは、汎用データ・ハンドラで、主にフレームの転送に用いられる。通常、GDHはそれぞれ専用制御プロセッサを持つ。制御プロセッサはそれぞれ、本発明に従って複数のコプロセッサを使用する。システムに必要なCLPの数は一般に、パフォーマンス評価により求められる。アーキテクチャと構造は完全にスケーラブルであり、シリコン領域によってのみ制限される。CLPの数が増えると、シリコン領域に含まれるアービタと命令メモリが大きくなる。

【００３２】管理セル・ハンドラ（GCH）のハードウェアはGDHと同じであるが、管理フレームはGCHによってのみ処理できる。GCHがデータ・フレームも処理するようになっている場合はWeb（CLP対応レジスタ）上でプログラミングできる（その場合GDHの役割を担う）。GCHは、ツリーの挿入や削除を行うため、GDHハードウェア・アシストにはないハードウェアを含む。GCHは、管理セル関連コードの実行、エージングのようなチップ、ツリー管理関連コードの実行、及びCPや他のGCHとの制御情報の交換に用いられる。そのような実行タスクがない場合、GCHはフレーム転送関連コードを実行し、その場合はGDHと全く同じように動作する。

【００３３】汎用プロセッサ・ハンドラ（GPH）。このプロセッサは、Power PCに接続されるハードウェア・メールボックスにアクセスする。RIT１／２にはPower PCはないので、GPHはGDHと全く同じように動作する。

【００３４】汎用ツリー・ハンドラ（GTH）には、ツリーの挿入、ツリーの削除、及びロープの管理を行うため、GDH及びGCHハードウェア・アシストにはないハードウェアがある。GTHは、GPQにツリー管理コマンドを含むフレームがないときデータ・フレームを処理する。

【００３５】図２を参照する。CLP＃１　３４は、命令フェッチ／デコード／実行ユニット８８、汎用レジスタ８０、及びスカラ・レジスタ８４とアレイ・レジスタ８６が含まれる専用レジスタ８２を含む。CLP＃２　３６も同種のコンポーネントを含む。

【００３６】CLP３４は、２つの命令を除いて、その実行ユニット１０２内で命令を完全に実行する。２つの例外は、図４の直接／間接コプロセッサ実行命令４１７である。これら２つの命令は、接続されたコプロセッサのうち１つでコマンド処理を開始する。コプロセッサはコマンドを互いに並行して実行でき、またCLP内の命令処理と並行して実行できる。CLP命令にコプロセッサがかかわるときは、コプロセッサ識別子と呼ばれ、操作のため選択されたコプロセッサを示す0乃至１５の範囲内の４ビットの数が指定される。

【００３７】共有メモリ・プール：
４Kバイトの共有メモリ・プール４４は、コプロセッサの少なくとも一部に対するアレイ・レジスタを保持する。PPUで動作する全てのスレッドにより用いられる。各スレッドにより１Kバイトが用いられ、次の領域に分けられる。FCBpage（エンキュー・コプロセッサのアレイ・レジスタと見なされる）、データ・フェッチ、スクラッチ・メモリ領域（CLPのアレイ・レジスタと見なされる）、及びシステム領域である。プールはスレッド数に応じて、等しいセグメントに分けられていると見なすことができる。各セグメントでは、アドレス・スペースがCLP及びアレイ・レジスタを必要とする各種コプロセッサのアレイ・レジスタに分けられる。プールのアドレス・ラインのうち２つは、どのCLPがアクティブか、どのスレッドがアクティブかに応じて駆動される。

【００３８】PPUコプロセッサ：コプロセッサはそれぞれ専用ハードウェア・アシスト・エンジンであり、コアに組み込まれた場合は、大量の直列化コードを必要とするような機能を実行する。コプロセッサはCLPと並

列に動作し、ＩＰヘッダの変更、フロー制御アルゴリズムに用いられるフロー情報の維持、ＣＡＢを介した内部レジスタへのアクセス、フロー制御及び管理情報ブロック（ＭＩＢ）のカウントの維持（標準とプロプライエタリ）、転送されるフレームのエンキュー等、データの移動に用いられる機能を提供する。プロセッサはそれぞれ、他に明記しない限り、ＰＰＵの各スレッドに対するスカラ・レジスタとアレイのセットを維持する。

【００３９】再び図２を参照する。ＰＰＵ１０は２つのコア言語プロセッサ３４、３６と接続された数個のコプロセッサ１２、２０、２２、２４、２６、２８、３０及び３２を含む。これらのコプロセッサは、高速パターン検索、データ操作、内部チップ管理機能、フレーム解析、及びデータ・フェッチ等、特定のネットワーク処理タスクについてハードウェア・アクセラレーションを実現する。

【００４０】以下、各種コプロセッサとその機能について説明する。

【００４１】ツリー検索コプロセッサ：ツリー検索エンジン（ＴＳＥ）コプロセッサ１２には、コプロセッサ識別子２が割当てられる。ＴＳＥは、ツリー管理とアービタ１１０を介したツリー検索メモリへの直接アクセスに関するコマンドを持つ。ＬＰＭ（可変長の一致を必要とする最長プレフィックス一致パターン）、ＦＭ（正確な一致のある固定サイズ・パターン）、及びＳＭＴ（範囲またはビット・マスク・セットを定義するパターンを伴うソフトウェア管理ツリー）の検索を行うアルゴリズムを持ち、フレームの転送、及び変更情報を取得する。コプロセッサ識別子１が割当てられるデータストア・コプロセッサ２６は、フレーム・データの収集、変更、またはネットワーク・プロセッサのフレーム・データ・メモリ１１２への導入に用いられる。本発明に有用なツリー検索のアーキテクチャや動作の詳細については、米国特許出願ドケット番号ＲＡＬ９９９０１３９、同ＲＡＬ９９９０１４０、及びＲＡＬ９９９０１４１を参照されたい。

【００４２】チェックサム・コプロセッサ：従来のチェックサム・コプロセッサ２０は、インターネット・チェックサムを計算するため提供されるアルゴリズムを使用してチェックサムを計算し検証する。その際、ハーフワード・データに対してチェックサム操作を実行し、ハーフワード・チェックサム結果を返す。次のコマンドを使用できる。

・チェックサム生成、及び
・チェックサム確認

【００４３】コマンドの結果は累算スカラ・レジスタとステイク・スカラ・レジスタに置かれる。累算スカラ・レジスタは、チェックサム計算の結果を格納し、ステイク・スカラ・レジスタは、チェックサムに含まれる最後のハーフワードに続くバイト位置を格納する。チェックサム・コプロセッサのデータは共有メモリ・プールに置かれる。

【００４４】コプロセッサに対するコマンドは次のオプションを含む。

１）ＩＰヘッダ：ＩＰヘッダが指示されたとき、レイヤ３ヘッダの開始位置（つまりステイク）が渡される。ハードウェアが、ヘッダ長フィールドからＩＰヘッダの長さを確認し、この値を長さスカラ・レジスタにロードする。チェックサムを生成する際、現在のチェックサムを格納したハーフワードの代わりに０の値が用いられる。

２）データ・ブロック：共有メモリ・プールにあるデータは、データのブロックとして処理され、チェックサムを生成または確認することができる。共有メモリ・プールの開始位置及び長さが渡される。データのブロックを確認する際、チェックサムは累算スカラ・レジスタに置かれる。データのブロックをチェックする際、チェックサムは累算レジスタに置かれている。

【００４５】エンキュー・コプロセッサ：エンキュー・コプロセッサ２４は２つの機能を提供する。

１）コードにより、ワーキングＦＣＢＰａｇｅと呼ばれ、アップ／ダウンＦＣＢページを作成するため用いられる２５６ビット・レジスタ、ＦＣＢ（フレーム制御ブロック）ページを作成することができる。レジスタは、フレームをＥＤＳ（エンキュー、デキュー／スケジューリング）アップまたはＥＤＳダウンでエンキューするため必要な全てのパラメータを格納する。レジスタに格納されるパラメータの例として、アップのＦＣＢアドレス、ターゲット・パート番号、フレーム変更情報、及び次ループＩＤ等がある。

２）ＣＬＰと完了ユニット（ＣＵ）のインタフェースを提供する。ＣＵはＣＬＰから独立して動作するが、ＣＬＰプロセッサ毎にＲｅａｄｙ　ＦＣＢＰａｇｅというレジスタを格納する。レジスタは、エンキューの後、ＣＵにコピーされ、その後、エンキューはエンキュー・プロセッサにより引き継がれる。その際、ＣＬＰが解放されて次のフレームが処理される。ＥＱはそのレディ・ビット（？）を設定する。ただし、ＣＵのＲｅａｄｙ　ＦＣＢＰａｇｅが空でない場合、ＥＱは、ＥＱレジスタが空になるまでＣＬＰからＣＵへの転送をブロックし、その後転送を可能にする。エンキュー・コプロセッサは、スレッドと完了ユニットの間のインタフェース及び共有メモリ・プールに維持されるＦＣＢＰａｇｅの使用を管理する。各スレッドに３つのＦＣＢＰａｇｅ位置があり、フレームに関するエンキュー情報をそこに維持することができる。ページのうち２つは、連続したエンキュー間で２つのページをスワップすることによって、完了ユニット・インタフェースに対するパフォーマンスを改良するため用いられる。スレッドに対して書かれるアセンブリ言語コードは、ハードウェアによって管理されるのでこれら２つのページを区別しない。３番目のページは、

コードにより新しいフレームを作成できるようにするためスレッドにより用いられる。この例として、学習のための管理トラフィック（guided traffic for learning）の作成がある。これはGTHスレッドにより実行されるように再エンキューされる。

【0046】CLPスレッドがエンキュー・コマンドを発行すると、FCBPageは使用中と指示される。他の位置が使用できる場合は、エンキュー・コプロセッサからの応答を待たずに新しいフレームがスレッドにディスパッチされる。完了ユニットは、エンキュー・コプロセッサを通して共有メモリ・プールからFCBPageをフェッチし、これをEDS（エンキュー・コマンドにより示される入口または出口）に提供する。これが起こるとFCBPageはフリーと指示される。両方のFCBPageが使用中と指示された場合、第3のフレームは起動できない。

【0047】エンキュー・コプロセッサでは次のコマンドがサポートされる。

・エンキュー入口（ENQUP）は、完了ユニットを介して入口フロー制御／スケジューラにエンキューする。

・エンキュー出口（ENQDN）は、完了ユニットを介して出口フロー制御／スケジューラにエンキューする。

・エンキュー・クリア（ENQCLR）は、現在のFCBPageをクリアする（全フィールドを0に設定する）。

【0048】データストア・コプロセッサ：このコプロセッサ26は次の機能を実行する。

1）アップ・データストア及びダウン・データストアとのインタフェースを取る。

2）タイマ・イベントのディスパッチまたは割込みのとき構成情報を受信する。

3）フレームのチェックサムを計算する。

このコプロセッサは通常、320バイトのデータ・バッファ及びそれぞれ128ビットのワード8個のメモリを含む。

【0049】フレーム・データはデータストア・コプロセッサを通してアクセスされ、メディアから受信されたフレームを格納する入口データストアと、パケット・ルーティング・スイッチから受信され再アセンブルされたフレームを格納する出口データストアとのインタフェースが取られる。また、タイマ・イベントのディスパッチや割込みのとき構成情報も受信される。

【0050】データストア・コプロセッサは、共有メモリ・プールで定義されたアレイを使用する。アレイはデータプールであり、8つのクォドワード及び2つのスクラッチ・アレイを保持でき、スクラッチ・アレイの1つは8つのクォドワード、もう1つは4つのクォドワードを保持する。データストア・コプロセッサには、入口、出口のデータストアとの間のアレイ内容の読取り、書込みの制御に用いられる別のスカラ・レジスタが維持される。データストア・コプロセッサによりサポートされるスレッド毎に、アレイとスカラ・レジスタの1セットが定義されている。

【0051】これら共有メモリ・プールのアレイは、データストア・コプロセッサの作業領域になる。データストアを直接読取りまたはそこに直接書込む代わりに、大量のフレーム・データがデータストアからこれら共有メモリ・プールのアレイに読取られるか、または大量のデータがこれらアレイからデータストアに書込まれる。転送単位はクォドワードであり、クォドワードは16バイトとして定義される。

【0052】データストア・コプロセッサには次のコマンドを使用できる。

1）出口データストア書込み（WRDNDS）：CLPにより出口データストアへの書込みができる。書込みはクォドワード単位の倍数でのみ発生する。データはデータストア・コプロセッサのアレイのいずれか（データプールまたはスクラッチ・アレイ）から取られる。

2）出口データストア読取り（RDDNDS）：CLPにより出口データストアからデータを読取り、データストア・コプロセッサのアレイの1つに入れることができる。読取りは、出口データストアに対して、クォドワード単位の倍数でのみ発生する。

3）入口データストア書込み（WRUPDS）：CLPにより入口データストアへデータを書込むことができる。読取りは、入口データストアに対して、クォドワード単位の倍数でのみ発生する。

4）入口データストア読取り（RDUPDS）：CLPにより入口データストアからデータを読取ることができる（クォドワード単位の倍数でのみ）。

5）出口データストアからの他のフレーム・データの読取り（RDMOREDN）：出口データストアからのハードウェア・アシスト読取り。RDMOREDNは、最後の読取りが停止したところからフレームの読取りを続け、データをデータプールに置く。データはデータプールに移されるので、ハードウェアは、読取られているフレームの現在位置を管理し、次のツイン・バッファの位置を確認するためツイン・バッファからリンク・ポインタをキャプチャする。このアドレスは、ツインが尽きて次のツインが読取られるまで、後続のRDMOREDN要求のためハードウェアにより用いられる。データプールの内容はツインの内容のマップなので、データプール内でフレーム・データがラップされる可能性がある。コードによりデータプール内のデータの位置が管理される。

6）入口データストアからの他のフレーム・データの読取り（RDMOREUP）：入口データストアからのハードウェア・アシスト読取り。RDMOREUPは、最後の読取りが停止したところからフレームの読取りを続け、データをデータプールに置く。データはデータプー

ルに移されるので、ハードウェアは、読取られているフレームの現在位置を管理し、フレームの次のデータ・バッファの位置を確認するためバッファ制御ブロック領域に維持されたリンクをキャプチャする。このアドレスは、データ・バッファが尽きて次のバッファが読取られるまで、後続のRDMOREUP要求のためハードウェアにより用いられる。コードによりデータプール内のフレームのデータの位置が管理される。

7）リース・ツイン・バッファ（LEASETWIN）：フリー・ツイン・バッファ（出口データストアで新しいデータを作成するときに用いられる）のアドレスを返す。

【0053】制御アクセス・バス（CAB）コプロセッサ：このコプロセッサ28では、ネットワーク・プロセッサが、ネットワーク・プロセッサ・システム全体で、選択されたレジスタを制御することができる。システム初期化等のために特定のレジスタを初期化でき、システム診断やメンテナンスのため特定のレジスタを読取ることができる。

【0054】コプロセッサは、組み込みプロセッサ複合体（EPC）Webアービタとインタフェースを取る。アービタはCLPとWebウォッチ間の調停を行う。これによりCLPは全てWeb上で読取り、書込みができる。

【0055】CABコプロセッサは、CLPスレッドに関してCABアービタと制御アクセス・バスにインタフェースを与える。スレッドは、CABのアドレス、データ等、CABアクセスのオペランドをロードする必要がある。その場合、CABにアクセスするプロトコルは、CABインタフェース・コプロセッサにより処理される。CABインタフェース・コプロセッサは次のコマンドを提供する。

・CABアクセス調停（WEBARB）：CABへのアクセスを取得するためスレッドにより用いられる。アクセスが取得されると、スレッドはCABを解放するまでCABの制御を維持する。

・CAB読取り／書込み（WEBACCESS）：CAB及びCABからアクセスできる接続されたレジスタとの間でデータを移動する。PPU内の送信元と宛先は汎用レジスタ（GPR）である。

・CAB優先使用（WEBPREEMPT）：GFHスレッドによってのみ用いられ、これによりGFHは、1回の読取り／書込みアクセスについて、CABがすでに他のスレッドに与えられている場合でもCABの制御を取得する。

【0056】チェックサム、データストア、エンキュー、CABの各コプロセッサのアーキテクチャ、動作に関する他の詳細については、整理番号RAL999008 3の米国特許出願、"String Copy (StrCopy) Coprocessor"を参照されたい。

【0057】ストリング・コピー・コプロセッサ22はCLPの機能を拡張し、データのブロックを移動する。データは共有メモリ・プール内でのみ移動する。次のコマンドが使用できる。

・ストリング・コピー（Strcopy）：このコマンドは、アレイ間でデータの複数のバイトを移動するため用いられる。コマンドは、ソース・データ・ブロックとシンク・データ・ブロックの開始バイト位置及び移動するバイト数を渡す。

【0058】カウンタ・コプロセッサ：カウンタ・コプロセッサ30は、全PPU間で共有できるカウンタ・マネージャ（図示せず）へのアクセスをバス72を通して提供する。コプロセッサは、全てのカウンタ・プログラムとインタフェースを取り、カウンタの更新を行う。スカラ・レジスタとコプロセッサ実行インタフェース16間のFIFOバッファ76で実装される。アレイ・レジスタとコプロセッサ・データ・インタフェース14間に第2FIFOバッファ78が置かれる。スレッドはそれぞれ自体のカウンタ・コプロセッサを持っているかのように動作する。このコプロセッサには外部（PPUに対して）アドレス／データ・バスが用いられる。これにより、外部バスを通してカウンタ・コプロセッサを使用するため2つ以上のPPUが調停を行えるようにシステムを実装することができる。

【0059】スレッドは、カウンタ・コプロセッサを通してカウンタの更新を要求し、カウンタ・マネージャが操作を完了するのを待たずに処理を続けることができる。カウンタ・コプロセッサはその要求をカウンタ・マネージャに通知し、カウンタ・アクセス・コマンドのオペランドを処理のためカウンタ・マネージャに渡す。カウンタ・コプロセッサには、PPUで動作する4つのスレッドにより発行されたカウンタ・アクセス・コマンドを入れる8ディープ・キューがある。カウンタ・コプロセッサは次のコマンドを提供する。

【0060】カウンタ・アクセス（CtrAccess）は、カウンタを増分するかまたはカウンタに値を追加する。コマンド・オペランドは、カウンタ識別子（カウンタ・メモリの形）、インデックスとオフセット、増分もしくは追加コマンド、値フィールド、カウントの読取りもしくは書込み、またはカウント値の読取りとクリアである。スレッドは、カウンタ・コプロセッサ・キューが一杯でなければ、コマンドの実行を待たない。

【0061】カウンタ・コプロセッサとその動作について詳しくは、整理番号RAL920000078US1の米国特許出願、"Coprocessor for Managing Large Counter Arrays"を参照されたい。

【0062】ポリシ・コプロセッサ：ポリシ・コプロセッサ32は、スレッドに関してポリシ・マネージャ（図示せず）とのインタフェース74を提供する。スカラ・レジスタとコプロセッサ実行インタフェース16間のF

ＩＦＯバッファ７６で実装される。アレイ・レジスタとコプロセッサ・データ・インタフェース１４間には第２ＦＩＦＯバッファ７８が置かれる。スレッドは、このインタフェースを通してフレームの"カラー"の更新を要求する。フレームのカラーは、ネットワーク・プロセッサの構成可能なフロー制御機構の一部として用いられ、この機構によりフレームに対する処理が決定される。スレッドはポリシ・マネージャが、ポリシ・コプロセッサを介して結果を返すまで待機する必要がある。ポリシ・マネージャは、このフレームがメンバーであるフローについてポリシ制御ブロックにアクセスする。オペランドには、ポリシ制御ブロック・アドレス、パケット長、フレームに現在割当てられているカラー等がある。返される結果はフレームの新しいカラーである。

【００６３】ＣＬＰ３４、３６はそれぞれ、コプロセッサ実行インタフェース１６とコプロセッサ・データ・インタフェース１４の2つのインタフェースを通してコプロセッサ１２、２０、２２、２４、２６、２８、３０及び３２に接続される。これらのインタフェースの機能については図4で詳しく説明する。

【００６４】ＰＰＵ内のコプロセッサはそれぞれ、４ビット・コプロセッサ識別子により識別される。各コプロセッサが最大２５６の専用レジスタをサポートする。コプロセッサ内の専用レジスタは、0乃至２５５の範囲の８ビット・レジスタ番号により識別される。コプロセッサ番号（ＣＰ＃）とレジスタ番号の組み合わせにより、ＰＰＵ内のレジスタが識別される。スカラ・レジスタとアレイ・レジスタの2種類の専用レジスタがある。

【００６５】レジスタ番号0乃至２３９はスカラ・レジスタに予約されている。スカラ・レジスタは最小1ビット、最大３２ビットである。スカラ・レジスタのビットには0乃至３１までの番号が振られ、0は右端またはＬＳＢ（least significant bit）、３１は左端またはＭＳＢ（most significant bit）である。３２ビット未満の長さのスカラ・レジスクは右揃えされ、残りのビットは非実装と見なされる。ＣＬＰが３２ビット未満の長さのスカラ・レジスタを読取るとき、非実装ビットの値はハードウェアに依存する。非実装ビットへの書込みは無効である。

【００６６】レジスタ番号２４０乃至２５５はアレイ・レジスタに予約されている。アレイ・レジスタは最小2バイト、最大２５６バイトである。ＣＬＰはアレイ・レジスタを読取るか書込み、共有メモリ・プール４４内で一度に2バイト（ハーフワード）、一度に4バイト（ワード）、または一度に１６バイト（クォドワード）パーティションをきる。

【００６７】汎用レジスクの使用方法は周知の通りであり、ここでは一般的なことについて述べる。プログラマから見た汎用レジスタは2通りある。汎用レジスクは、０、２、４．．．．１４の集合から4ビット数で表される３２ビット・ラベルで示されるように、３２ビット・レジスタと見なすこともできる。この意味でプログラマは8個の３２ビット汎用レジスタを扱う。またプログラマは汎用レジスタを、０、１、２、．．．１５の集合からの4ビット数として表される１６ビット・ラベルに従って、１６ビット・レジスタとして扱うこともできる。この意味でプログラマは１６個の１６ビット・レジスクを扱う。

【００６８】各コプロセッサに、ビジー信号フィールドからの情報を格納するステータス・レジスタが含まれる。このレジスタは、所与のコプロセッサが利用できるか、またはビジー状態かどうかをプログラマに示す。コプロセッサ完了コード・レジスタが図４のＯＫ／Ｋ、Ｏ．フィールド４１５からの情報を格納する。従って、プログラマは、所与のコプロセッサがビジーかまたは利用できるか知る必要のある場合。この情報をコプロセッサのステータス・レジスタから取得することができる。同様にコプロセッサ完了コード・レジスタが、コプロセッサ・タスクの完了についてプログラマに情報を提供する。

【００６９】各ＣＬＰに次の１６ビット・プログラム・レジスタが含まれる。プログラム・カウンタ・レジスタ、プログラム・ステータス・レジスタ、リンク・レジスタ、及びキー長レジスタである。タイムスタンプ・レジスタと乱数ジェネレータ・レジスタの２つの３２ビット・レジスタも追加される。前記レジスタそれぞれにスカラ・レジスタ番号も与えられる。

【００７０】プログラマから見た汎用レジスタは2通り考えられる。プログラマは汎用レジスタを３２ビット・レジスクと見なすことができ、１６ビット・レジスクと見なすこともできる。

【００７１】アレイ・レジスタは、アレイ・レジスタ番号を通してプログラマに知られる。

【００７２】図4は、コプロセッサ実行インタフェース１６とコプロセッサ・データ・インタフェース１４を通してＣＬＰ３４をそのコプロセッサ４０１に接続するインタフェース信号を示す。個々のワイヤ接続数は、個々の割当てアイテムの矢印の横にある番号ラベルに示してある。ここの説明では、選択コプロセッサ２０、２２、．．．は、コプロセッサ識別子が、後述する操作に応じて４１１、４２０、または４２９に現れるコプロセッサ識別子に一致するコプロセッサを表す。

【００７３】ＣＬＰ３４は、実行インタフェース１６により、任意のコプロセッサ２０、２２、．．．上でコマンドの実行を開始することができる。コプロセッサ番号４１１は、コマンドのターゲットとして１６のコプロセッサのうち１つを選択する。ＣＬＴにより開始フィールド４１０が論理"1"になると、コプロセッサ番号４１１により示される選択コプロセッサ４５０が、６ビットＯｐフィールド４１２により指定されたコマンドの実行を

開始する。Op引数４１３は、４４ビットのデークで、コプロセッサ４５０により処理されるようにコマンドとともに渡される。ビジー信号４１４は１６ビット・フィールドで、各コプロセッサ４０１に１ビットであり、コプロセッサがコマンドを実行していてビジーか（ビット＝１）または、そのコプロセッサがコマンドを実行していない（ビット＝０）ことを示す。これら１６ビットはスカラ・レジスタに保存され、レジスタのビット０はコプロセッサ０に、ビット１はコプロセッサ１に、以下同様に対応する。ＯＫ／Ｋ．Ｏ．フィールド４１５は１６ビット・フィールドで、各コプロセッサ４０１に１ビットである。これは１ビット戻り値コードであり、コマンドに依存する。例えば、コプロセッサ４０１に与えられたコマンドが失敗に終わったか、コマンドが成功したかをＣＬＰ３４に示すためこれを使用できる。この情報はＣＬＰスカラ・レジスタ内に保存され、レジスタのビット０はコプロセッサ０に、ビット１はコプロセッサ１に、以下同様に対応する。直接／間接フィールド４１７は、コプロセッサ実行命令のどのフォーマットが実行されているかを選択コプロセッサ４５０に示す。直接／間接＝０のとき、直接フォーマットが実行されている。直接／間接＝１のときは間接フォーマットが実行されている。

【００７４】コプロセッサ・データ・インタフェース１４は３つの信号グループを含む。書込みインタフェース４１９、４２０、４２１、４２２、４２３、４２４は、コプロセッサ内のスカラ・レジスタまたはアレイ・レジスタにデータを書込むときに関係する。読取りインタフェース４２７、４２８、４２９、４３０、４３１、４３２、４３３は、コプロセッサ内のスカラ・レジスタ８４またはアレイ・レジスタ８６のいずれかの専用レジスタ８２からデータを読取るときに関係する。第３のグループ４２５、４２６、４２７は、スカラ・レジスタまたはアレイ・レジスタの読取りと書込みの両方で用いられる。読取りインタフェースと書込みインタフェース両方に対する複製機能は、レジスタからレジスタへデータを移動するための同時読取り／書込みをサポートするように働く。

【００７５】書込みインクフェースは、書込みフィールド４１９を使用して、コプロセッサ番号４２０により示されるコプロセッサ４５０を選択する。書込みフィールド４１９は、ＣＬＰ３４が選択コプロセッサにデータを書込もうとするときに１に設定される。コプロセッサ・レジスタ識別子４２１は、ＣＬＰ３４が選択コプロセッサ内４５０に書込もうとすることをレジスタに示す。コプロセッサ・レジスタ識別子４２１は８ビット・フィールドで、よって２５６のレジスタがサポートされる。０乃至２３９の範囲のコプロセッサ・レジスタ識別子はスカラ・レジスタへの書込みを示す。２４０乃至２５５の範囲のコプロセッサ・レジスタ識別子はアレイ・レジスタへの書込みを示す。アレイ・レジスタ書込みの場合、オフセット・フィールド４２２は、アレイ・レジスタのデータ書込み操作の開始点を示す。このフィールドは８ビットで、従ってアレイ内で２５６のアドレスをサポートする。データ出力フィールドは、コプロセッサ４５０に書込まれるデータを入れる。これは１２８ビットの大きさで、従って最大１２８ビットの情報を一度に書込むことができる。書込み有効フィールド４２４は、コプロセッサ４５０がデータの受信をいつ終了したかをＣＬＰ３４に示す。これによりＣＬＰ３４は、コプロセッサ４５０がデータを取る間、一時停止し、データを有効な状態に保つことができる。

【００７６】読取りインタフェース１４は、書込みインタフェース１６と構造は似ているが、データはコプロセッサから読取られる。読取りフィールド４２８は書込みフィールド４１９に対応し、選択コプロセッサ４５０で読取り操作がいつ実行されるか示すためＣＬＰ３４により用いられる。コプロセッサ番号識別子フィールド４２９は、どのコプロセッサ４５０が選択されているかを示す。レジスタ番号フィールド４３０、オフセット・フィールド４３１、及び読取り有効フィールド４３３は、書込みインタフェースの４２１、４２２及び４２４に対応する。データ入力フィールド４３２は、コプロセッサ４５０からＣＬＰ３４へのデータを入れる。

【００７７】読取りまたは書込みの操作は、３つの長さのうちいずれか１つをとる。１６ビットが転送されることを示すハーフワード、３２ビットが転送されることを示すワード、及び１２８ビットが転送されることを示すクォドワードである。読取りデータ４３２と書込みデータ４２３は１２８ビット幅である。１２８ビット未満のデータ転送は右揃えされる。信号４２５及び４２６は、データ転送サイズを示す。１６ビットの転送は４２５と４２６が両方とも０で示され、３２ビットの転送は４２５と４２６がそれぞれ１と０で示され、１２８ビットの転送は４２５と４２６がそれぞれ０と１で示される。

【００７８】修飾フィールド４２７は、データ読取りまたはデータ書込みの操作で用いられる。コプロセッサはそれぞれ、コプロセッサのハードウェア・デザイナにより定義された独自の方法でその意味を解釈する。これによりプログラマは、読取りまたは書込みの操作のときにハードウェアに対して情報ビットを追加指定することができる。データストア・コプロセッサは、パケット・バッファのリンク・リストでパケット・バッファのリンク・フィールドを省略することができる。

【００７９】コプロセッサでタスクを開始した後、ＣＬＰは、命令の実行を続けるか、またはコプロセッサでタスクが完了するまで実行を中断することができる。ＣＬＰが、コプロセッサ内のタスク実行と並行して命令の実行を続ける場合、後の時点で、メイン・プロセッサ・ユニットによるＷＡＩＴ命令の実行のため、１つ以上のコ

プロセッサでのクスク実行が完了するまで他の命令の実行が中断する。ＷＡＩＴ命令は、形式によっては１つ以上のコプロセッサ内でタスクが完了するまでＣＬＰ上の実行を中断させる。その時点でＣＬＰは命令の実行をＷＡＩＴ命令に続く命令から再開する。他の形では、ＷＡＩＴ命令により、特定のコプロセッサ内でタスクが完了するまでＣＬＰの実行が中断する。そのタスクが完了すると、ＣＬＰはコプロセッサからの１ビット戻りコードを、ＷＡＩＴ命令からの１ビットとともに調べ、ＷＡＩＴ命令に続く命令から命令の実行を再開するか、実行をプログラマにより指定された他の命令に分岐するか確認する。

【００８０】コプロセッサ実行命令は、コプロセッサでのコマンド処理を真似るため、図１のコプロセッサ実行インタフェース１６の"開始"信号を１に設定する。図５乃至８を参照する。コプロセッサ識別子５２０は、命令フィールド５００から取得され、開始信号を介して選択コプロセッサを示す。６ビット・コプロセッサ・コマンドは命令フィールド５０１から取得され、どのコマンドの実行を開始するかを信号により選択コプロセッサに示す。開始信号がアクティブにされ１になると、選択コプロセッサはそのビジー信号をアクティブにし（１にする）、コマンドの実行を完了するまで１のままにしておく。コマンドの実行が完了すると、この信号を０にする（非アクティブ化）。ＣＬＰは継続して１６ビットの信号を読取り、それらをそのスカラ・レジスタに入れる。コマンドの完了後、選択コプロセッサはこのステータスをスカラ・レジスタに入れる。

【００８１】再び図５乃至図８を参照する。命令の非同期実行フィールド５０２が０のとき、ＣＬＰはコマンドの完了を示すため、そのビジー信号を無効化する。そのとき、ＣＬＰは命令のフェッチと実行を再開する。命令の非同期実行フィールド５０２が１のとき、ＣＬＰは、ビジー信号の状態にかかわらず命令のフェッチと実行を続ける。

【００８２】選択コプロセッサでコマンド処理が開始されると、ＣＬＰは４４ビットの他のコマンド対応情報を信号によりコプロセッサに与える。この情報は、図５乃至図８に示すように命令フォーマットに応じて４つの方法のいずれかで引き出される。

【００８３】図５のコプロセッサ実行間接フォーマットは、上位１２ビット５２３のコマンド情報を命令フィールド５０４から取得する。下位３２ビットのコマンド情報５２４は３２ビット汎用レジスタ５０５から取得される。選択レジスクは、値｛０，２，４，．．．，１４｝に制限された４ビット命令フィールド５０３により求められる。こうしてそのレジスタから３２ビット・レジスタが選択される。ＣＬＰは信号を１に設定し、これが命令の間接フォームであることを選択コプロセッサに示す。

【００８４】実行命令の条件付きコプロセッサ実行間接フォーマットを図６に示す。ここで命令は、満足した特定の条件をもとに図５と同じように実行される。条件を満足しない場合、命令は実行されない。命令は、ＣＬＰのＡＬＵコードをもとに実行される。条件付き実行には４ビットが使用され、その結果Ｏｐフィールドは２ビットに短縮される。従って考えられる６４のコマンドのうち４つに対して条件付き実行が可能になる。他のコマンドは０と見なされる。よって長い待ち時間と短い待ち時間をもとにした条件付き実行を実現することができる。コプロセッサ命令の条件付き実行は特に、カウンタ・コプロセッサの動作と関連するときに有益である。

【００８５】図７のコプロセッサ実行直接フォーマットは、下位１６ビット５２７のコマンド情報を命令フィールド５０６から取得する。上位２８ビット５２６のコマンド情報は０に設定される。ＣＬＰは信号を０に設定し、これが命令の直接形式であることを選択コプロセッサに示す。

【００８６】条件付きコプロセッサ実行直接フォーマットを、図７と同じように実行されるように構成された形で図８に示す。図６と同様、条件付き実行では、４ビットが使用され、その結果Ｏｐフィールドは２ビットに短縮される。よって、考えられる６４のコマンドのうち４つに対して条件付き実行が可能になる。

【００８７】図９は、コプロセッサ待機命令のフォーマットを示す。ＣＬＰは、コプロセッサ・ステータス・レジスタで、命令フィールド６００から取得された１６ビット・マスクに対してビット単位でＡＮＤ演算を行う。結果が０でない、つまり１つ以上のコプロセッサが現在まだコマンドを実行している場合、ＣＬＰは命令のフェッチと実行を中断する。ただし、前記ＡＮＤ演算の実行は、結果が０になるまで継続する。

【００８８】図１０は、コプロセッサ待機／分岐フォーマットを示す。コプロセッサ識別子フィールド６０１は、コプロセッサ・ステータスの特定のビットがテストされることを示す。例えばフィールド６０１に１があるとき、コプロセッサ・ステータス・レジスタのビット１がテストされる。識別子フィールド６０１に１５があるとき、コプロセッサ・ステータスのビット１５がテストされる。テストされるビットの値が１で、対応するコプロセッサがコマンドの実行をまだ完了していないことを示す場合、ＣＬＰは命令のフェッチと実行を中断する。ただし前記の演算はテスト・ビットの値が０、つまり対応するコプロセッサがコマンドの実行を完了するまで継続する。この時点で、命令のＯＫフィールド６０２の値と、コプロセッサ識別子６０１により選択された、スカラ・レジスタのコプロセッサ完了コードのビットの値に応じて、２つの処理のうちいずれかが発生する。ＣＬＰは、下表に応じて、次のシーケンシャル命令のフェッチと実行を再開するか、または分岐して、命令フィールド６０３により示される命令アドレスから命令のフェッチ

と実行を再開する。

【表1】

６０２の値	選択コプロセッサ完了コード・ビットの値＝０	選択コプロセッサ完了コード・ビットの値＝１
０	分岐	次の命令
１	次の命令	分岐

【００８９】コプロセッサ・ユニットでタスクを開始するときの命令の実行については、米国特許出願第５４８１０９号、"Coprocessor Structure and Method for a Communications System Network Processor"を参照されたい。

【００９０】本発明は、更に、各ＣＬＰの複数の命令実行スレッド（それぞれ、処理中の別々のパケットに関係する）それぞれの独立したプロセス及びデータ・アクセス時の待ち時間の処理に関する。実行スレッドはそれぞれ独立したプロセスであり、スレッドがコプロセッサ・ハードウェアにアクセスできるときに命令のシーケンスを実行する。ツリー検索コプロセッサは、パイプラインにされ、ツリー検索パイプラインで複数の実行スレッドがそれぞれ同時に、ただし異なるフェーズで（オーバラップして）アクセスすることができる。本発明は、好適には、オーバヘッド０で複数の命令実行スレッドを採用し、実行をスレッドからスレッドに切り替える。スレッドはキューにされ、共有メモリに対するアクセスが高速に配信される。スレッドをキューにすることで、長い待ち時間のイベントに対して優先順位が最大のスレッドを可能な限り速く得ることができる。

【００９１】前述のように、ＰＰＵはそれぞれ、実行スレッド毎に１つ、複数の命令プリフェッチ・バッファを含む。これらのプリフェッチ・バッファにより、アクティブな実行スレッドにより命令の帯域幅が十分利用されていないインターバル時、アイドル中の実行スレッドのため命令のプリフェッチが可能になる。これにより、制御が新しい実行スレッドに切り替わったとき、そのスレッドの命令プリフェッチ・バッファが一杯になりやすく、よって、実行に利用できる命令が足りないために新しいスレッドがすぐ中断する可能性がなくなる。こうして、命令メモリに対するアクセス優先順位は、現在実行中のスレッドが最大の優先順位になり、現在のスレッドが中断したときに制御を取る立場にある実行スレッドが第２の優先順位を与えられるように制御される。同様に、実行キューの下端にある実行スレッドには、命令フェッチ・アクセスで最後の優先順位が与えられる。

【００９２】待ち時間の長いイベントのため（ツリー検索等）、アクティブなスレッドの実行が中断したとき、次のスレッドにフル制御が与えられるか、または待ち時間の短いイベント（ローカル・データ・ストレージでのコプロセッサの操作もしくは命令フェッチ待ち時間）のため実行が中断したときは、一時的制御が次のスレッドに与えられる。一時的制御が他のスレッドに与えられた場合、制御は、ブロックが解除されるとすぐに元のスレッドに返される。逆に他のスレッドにフル制御が与えられた場合、その他のスレッドは、ブロックされるまで制御を保つ。これにより待ち時間の短いイベントでのサイクルの無駄が避けられるが、１次実行スレッドが待ち時間の長いイベントに届くまでの時間は短くなる。他の場合、複数の実行スレッドがほぼ同じ時間に待ち時間の長いイベントに届く可能性があり、１つのスレッドのＰＰＵ実行を他のスレッドのツリー検索とオーバラップするメリットは少なくなる。待ち時間をもとに制御を割当てることについて詳しくは、整理番号ＲＡＬ９２０００００８の米国特許出願第５４２１８９号、"Network Processor with Multiple Instruction Threads"を参照されたい。この割当てとスレッド実行制御の詳細は次のようになる。

【００９３】コンピュータの電源が初めて投入されたとき、各ＣＬＰスレッドは初期化状態にある。パケットがプロセッサにディスパッチされると、対応するスレッドがレディ状態に変わり、その時点で実行サイクル要求を開始する。

【００９４】アービタが、アービタの論理関数をもとにプール式に従って実行サイクルをスレッドに与える。サイクルが与えられた場合、スレッドはレディ状態から実行状態に移る。実行状態のスレッドは、待ち時間イベントまたは処理されているパケットがエンキューされ、よってそのパケットに対するコードの作業が終了したことが示されたため実行が中断するまで要求を出し続ける。サイクルが与えられなくなると、これは他のスレッドが制御を取ったことを示す。これは、アービタがサイクルを与えない唯一の理由である。ただし、これら２つの状態のいずれか（レディまたは実行）で、スレッドは、パケット処理が終わりに達し、次のパケットがディスパッチのためキューに入るまで、新しい実行サイクルを要求し続け、待ち時間イベントで一時停止する。システムはそこで初期化状態に戻り、次のパケットを待機する。待機状態は、待ち時間の長いイベントまたは短いイベントを扱う。どのイベントが発生するかにかかわらず、プロセッサは中断し、アクティブなスレッドはデフォルトで待機状態になる。スレッドはそこで、待ち時間イベントが完了するまで実行サイクルの要求を停止する。

【００９５】スレッドを初期化状態からレディ状態に移すのと同じディスパッチ操作により、スレッド番号がＦＩＦＯバッファに入り、第１パケットがディスパッチされるスレッドは、優先順位が最高のスレッドになる。後のディスパッチ操作では、他のスレッド番号がＦＩＦＯに送られる。ＦＩＦＯの優先順位が最大のスレッド番号

は、待ち時間の長いイベントに出会うまでその位置にとどまり、その後、スレッドはＦＩＦＯの始めに戻り、最大優先順位から最小優先順位のスレッドに変わる。待ち時間の短いイベントによってスレッドがＦＩＦＯバッファで優先順位を失うことはない。

【００９６】スレッドがパケットの処理を終了すると、パケットは、出力ポートに転送するためエンキューされ、スレッド番号がＦＩＦＯバッファから移される。

【００９７】新しいパケットは、ハイレベル・コントローラ（図示せず）からディスパッチされる。このコントローラはスレッドとプロセッサを選択して各パケットを処理する。この決定により入力コマンドがＦＩＦＯバッファに送られる。また入力が状態機械に送られ、初期化状態からレディ状態に移行することが状態機械に指示される。外部コントローラからのそのコマンドとともに、パケットのディスパッチ先であるスレッド番号もコントローラからＦＩＦＯに送る必要がある。

【００９８】基本的に、実行を中断させるイベントは現在のプログラムの流れで短い割込みになるイベントと長い割込みになるイベントの２種類ある。短い割込みは、プログラムの流れが変わったため命令プリフェッチ・キューを再び埋める必要のある分岐命令により発生することがある。或いはまた、プログラムはコプロセッサがプロセッサのローカル・メモリでデータ関連タスクを実行するのを待機している間に中断することがある。この例は、チェックサム・コプロセッサが、変更されたヘッダ・フィールドで新しいチェックサムを計算する場合である。待ち時間が２５プロセッサ・サイクル未満のとき、イベントは、短い割込みと見なされる。待ち時間の長いイベントは通常、２５を超える待ち時間を伴い、通常は５０プロセッサ・サイクル乃至１００プロセッサ・サイクルを超える。これらは全体のパフォーマンスに大きな影響を与える。

【００９９】待ち時間の長いイベント、短いイベントを確認する手段は他に多数ある。待ち時間の長さはプログラマが制御でき、その場合ハードウェアやその構成は確認の際の要素にはならない。一方、しきい値レジスタを２５サイクルのしきい値で設定することもでき、その場合、操作に必要になるサイクル数はハードウェアにより確認され、その確認をもとに自動的な判断が行われる。

【０１００】コプロセッサ命令は、プロセッサが実行する命令の１タイプである。フィールドのビットの一部は、どのコプロセッサが対象かを示す。１ビットにより、特定の命令が待ち時間の長いイベントまたは短いイベントとして定義される。従って、プログラマは、同じアクセスを２つ定義することができる。１つは待ち時間の長いイベントとして、もう１つは待ち時間の短いイベントとして定義される。スレッド実行制御関数は、こうした待ち時間の長いイベントの影響を最小にするために設計されている。よって待ち時間の長いイベントにより、フル制御が別の実行スレッドに切り替わり、待ち時間の短いイベントにより一時的にのみ他のスレッドへの切り替えが起こる。

【０１０１】プロトコル・プロセッサ・ユニット（ＰＰＵ）とコア言語プロセッサの詳細は、当業者には周知の通りであり、本発明の一部を構成しないが、それらは、変更或いは実装することでネットワーク・プロセッサ・システム全体のアーキテクチャの一部になっており、特定の機能コプロセッサやシステムの他のコンポーネントと連携する。本発明で有用な個々のコプロセッサのアーキテクチャやプログラミングを含めた詳細は、本発明の一部を構成するものと見なされない。

【０１０２】まとめとして、本発明の構成に関して以下の事項を開示する。

【０１０３】（１）ネットワーク・プロセッサのプログラミング機能を制御する組み込みプロセッサ複合体のオペレーションであって、該プロセッサ複合体は、複数のプロトコル・プロセッサ・ユニット（ＰＰＵ）を含み、各ＰＰＵは少なくとも１つのコア言語プロセッサ（ＣＬＰ）を含み、各ＣＬＰは少なくとも２つのコード・スレッドを持ち、各ＰＰＵはＰＰＵの特定のタスクを実行する上で有用な複数のコプロセッサ及び複数の論理コプロセッサ・インタフェースを利用し、各ＣＬＰと該コプロセッサ間のアクセスを実現する、オペレーション。

（２）前記コプロセッサは、各ＣＬＰの複数のコード・スレッドをサポートする専用コプロセッサを含む、前記（１）記載のオペレーション。

（３）前記コプロセッサは、ツリー検索コプロセッサ、チェックサム・コプロセッサ、ストリングコピー・コプロセッサ、エンキュー・コプロセッサ、データストア・コプロセッサ、ＣＡＢコプロセッサ、カウンタ・コプロセッサ、及びポリシ・コプロセッサを含むグループから選択される、前記（１）記載のオペレーション。

（４）複数のスレッド間の優先順位を確認するためコプロセッサ実行インタフェース・アービタを含む、前記（３）記載のオペレーション。

（５）データ・スレッド間の優先順位を確認するためコプロセッサ・データ・インタフェース・アービタを含む、前記（３）記載のオペレーション。

（６）各スレッドと少なくとも１つのコプロセッサ間にＦＩＦＯバッファを含む、前記（３）記載のオペレーション。

（７）前記ＦＩＦＯバッファは各スレッドと前記カウンタ・コプロセッサの間にある、前記（６）記載のオペレーション。

（８）前記ＦＩＦＯバッファは各スレッドと前記ポリシ・コプロセッサの間にある、前記（６）記載のオペレーション。

（９）ネットワーク・プロセッサのプログラミング機能を制御する組み込みプロセッサ複合体を含むネットワー

ク処理システムであって、該複合体は複数のプロトコル・プロセッサ・ユニット（ＰＰＵ）を含み、各ＰＰＵは、それぞれ少なくとも２つのコード・スレッドを持つ少なくとも１つのコア言語プロセッサ（ＣＬＰ）と、前記システムの特定のタスクを実行する複数のコプロセッサ及び該コプロセッサのリソースにアクセスし各ＣＬＰと共有する複数のコプロセッサ・インタフェースと、を含む、システム。

（１０）前記コプロセッサ・インタフェースは、各ＣＬＰのコード・スレッドをサポートすることにのみ用いられる、前記（９）記載のネットワーク処理システム。

（１１）前記コプロセッサは、ツリー検索コプロセッサ、チェックサム・コプロセッサ、ストリングコピー・コプロセッサ、エンキュー・コプロセッサ、データストア・コプロセッサ、ＣＡＢコプロセッサ、カウンタ・コプロセッサ、及びポリシ・コプロセッサを含むグループから選択される、前記（１０）記載のネットワーク処理システム。

（１２）各スレッドと前記コプロセッサのうち少なくとも１つの間にＦＩＦＯバッファを含む、前記（１０）記載のネットワーク処理システム。

（１３）前記ＦＩＦＯバッファは各スレッドと前記カウンタ・コプロセッサの間にある、前記（１２）記載のネットワーク処理システム。

（１４）前記ＦＩＦＯバッファは各スレッドと前記ポリシ・コプロセッサの間にある、前記（１２）記載のネットワーク処理システム。

（１５）前記ＣＬＰのスレッドにより実行される特定の操作命令を含み、該実行の結果、コプロセッサ・オペレーションを制御するコマンドが得られ、該コマンドは前記ＣＬＰとコプロセッサ間のインタフェースを流れる、前記（９）記載のネットワーク処理システム。

（１６）前記命令は、特定のコプロセッサ・オペレーションの条件付き実行を可能にするように働く、前記（１５）記載のネットワーク処理システム。

（１７）前記命令により、前記システムが、特定のコプロセッサ・コマンドに応答してデータにアクセスするための予測応答時間に従って、待ち時間の長いイベントと待ち時間の短いイベントを識別し、アクティブなスレッドの実行が待ち時間の長いイベントにより中断したときに他のスレッドにフル制御を与えるか、またはアクティブなスレッドの実行が待ち時間の短いイベントにより中断したときに他のスレッドに一時的制御を与える、前記（１５）記載のネットワーク処理システム。

（１８）複数のプロトコル・プロセッサ・ユニット（ＰＰＵ）を含む組み込みプロセッサ複合体内の命令の実行を制御する方法であって、該プロトコル・プロセッサ・ユニットはそれぞれ少なくとも１つのコア言語プロセッサ（ＣＬＰ）を含み、該ＣＬＰはそれぞれ少なくとも２つのコード・スレッドを持ち、該方法は、該ＰＵに対する特定のタスクを実行するため、各ＰＰＵによる複数のコプロセッサの使用、及び該コプロセッサと各ＣＬＰ間のアクセスを提供する複数の論理コプロセッサ・インタフェースの使用を含む、方法。

（１９）前記ＰＰＵの複数のコード・スレッドをサポートする専用コプロセッサの使用を含む、前記（１８）記載の方法。

（２０）前記コプロセッサの1つ以上は、ツリー検索コプロセッサ、チェックサム・コプロセッサ、ストリングコピー・コプロセッサ、エンキュー・コプロセッサ、データストア・コプロセッサ、ＣＡＢコプロセッサ、カウンタ・コプロセッサ、及びポリシ・コプロセッサを含むグループから選択される、前記（１９）記載の方法。

（２１）実行スレッド間の優先順位を確認するためコプロセッサ実行インタフェース・アービタが用いられる、前記（２０）記載の方法。

（２２）データ・スレッド間の優先順位を確認するためコプロセッサ・データ・インタフェース・アービタが用いられる、前記（２０）記載の方法。

（２３）各スレッドと前記コプロセッサのうち少なくとも１つの間にＦＩＦＯバッファを提供するステップを含む、前記（２０）記載の方法。

（２４）前記ＦＩＦＯバッファは各スレッドと前記カウンタ・コプロセッサの間にある、前記（２３）記載の方法。

（２５）前記ＦＩＦＯバッファは各スレッドと前記ポリシ・コプロセッサの間にある、前記（２３）記載の方法。

（２６）前記ＣＬＰにより実行される特定の操作命令を提供するステップを含み、該実行の結果、コプロセッサ・オペレーションを制御するコマンドが得られ、該コマンドは前記ＣＬＰとコプロセッサの間のインタフェースを流れる、前記（１８）記載の方法。

（２７）前記操作命令により特定のコプロセッサ・オペレーションの条件付き実行が可能になる、前記（２６）記載の方法。

（２８）前記実行は直接的または間接的である、前記（２７）記載の方法。

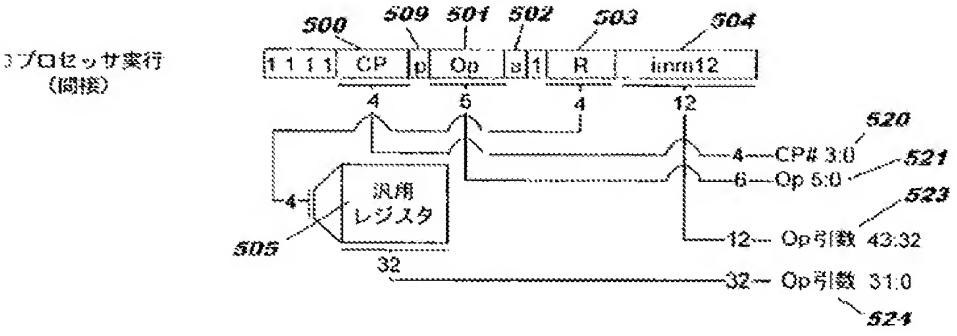
（２９）前記システムが、特定のコプロセッサ・コマンドに関する予測応答時間に従って、待ち時間の長いイベントと待ち時間の短いイベントを識別し、アクティブなスレッドの実行が待ち時間の長いイベントにより中断したときに他のスレッドにフル制御を与えるか、またはアクティブなスレッドの実行が待ち時間の短いイベントにより中断したときに他のスレッドに一時的制御を与える、命令を提供するステップを含む、前記（１８）記載の方法。

【図面の簡単な説明】

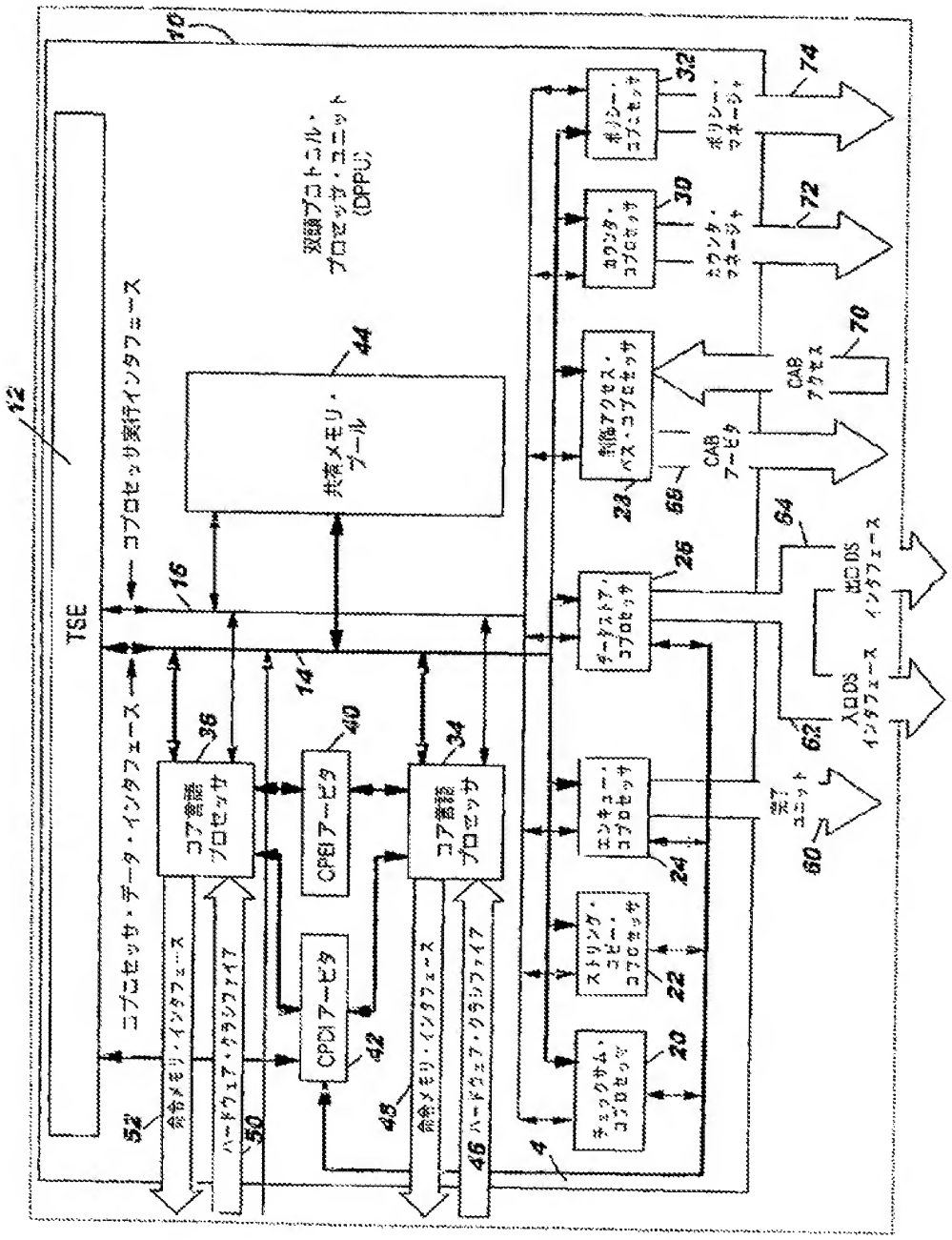
【図１】２つのコア言語プロセッサとコプロセッサを持つプロトコル処理ユニットを示す図である。

【図２】２つのコア言語プロセッサとコプロセッサがインタフェースを取るプロトコル処理ユニットを示す図である。
【図３】コア言語プロセッサと選択されたコプロセッサのインタフェースを示す図である。
【図４】コプロセッサ実行インタフェースとコア言語プロセッサを複数のコプロセッサに接続するコプロセッサ・データ・インタフェースを示す図である。
【図５】コプロセッサ実行命令フォーマットを示す図である。
【図６】コプロセッサ実行命令フォーマットを示す図である。
【図７】コプロセッサ実行命令フォーマットを示す図である。
【図８】コプロセッサ実行命令フォーマットを示す図である。
【図９】コプロセッサ待機命令フォーマットを示す図である。
【図１０】コプロセッサ待機命令フォーマットを示す図である。
【符号の説明】
１０　プロトコル・プロセッサ・ユニット（ＰＰＵ）
１２　ツリー検索エンジン
１４　データ・インタフェース
１６　実行インタフェース
２０　チェックサム・コプロセッサ
２２　ストリング・コピー・コプロセッサ
２４　エンキュー・コプロセッサ
２６　データストア・コプロセッサ
２８　制御アクセス・バス・コプロセッサ
３０　カウンク・コプロセッサ
３２　ポリシ・コプロセッサ
３４、３６　コア言語プロセッサ（ＣＬＰ）
４０　コプロセッサ実行インタフェース（ＣＰＥ）アービタ
４２　コプロセッサ・データインタフェース（ＣＰＤＩ）アービタ
４４　共有メモリ・プール
５６　命令メモリ
６８　ＣＡＢアービタ
７４　インタフェース
７６、７８　ＦＩＦＯバッファ
８０　汎用レジスタ
８２　専用レジスタ
８４　スカラ・レジスタ
８６　アレイ・レジスタ
８８　命令フェッチ／デコード／実行ユニット
１０２　実行ユニット
１１０　ツリー検索メモリ（ＴＳＭ）アービタ
１１２　フレーム・データ・メモリ
４０１、４５０　コプロセッサ
４１３　Ｏｐ引数
４１４　ビジー信号
４１９、４２０、４２１、４２２、４２３、４２４　書込みインタフェース
４２１　コプロセッサ・レジスタ識別子
４２７、４２８、４２９、４３０、４３１、４３２、４３３　読取りインタフェース
４５０　選択コプロセッサ
５０５　３２ビット汎用レジスタ
５２０　コプロセッサ識別子
５２３　上位１２ビット
５２４　コマンド情報

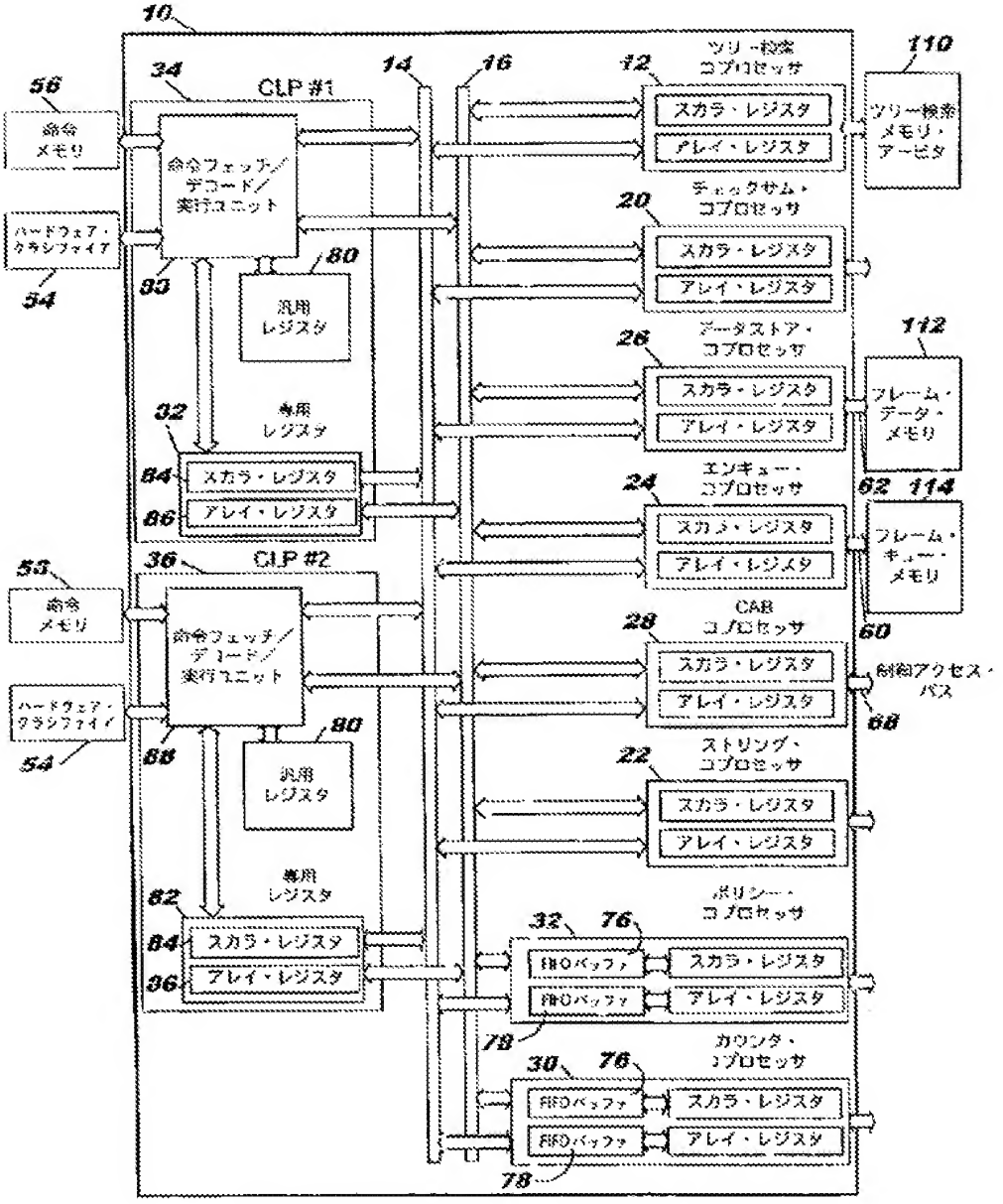
【図５】



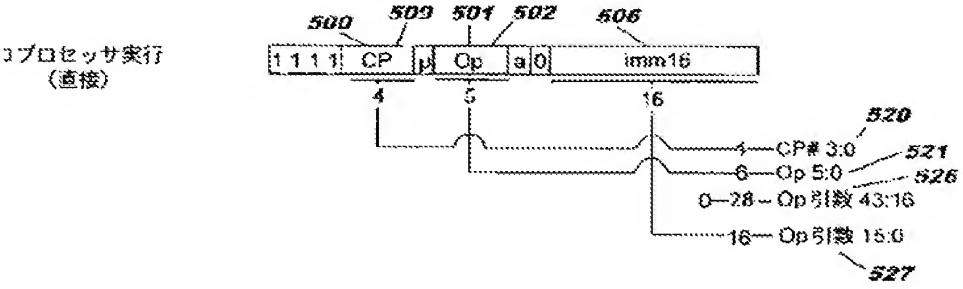
【図1】


10
12
TSE
コプロセッサ実行インタフェース
16
14
共有メモリ・プール
44
双線プロトコル・プロセッサ・ユニット (DPPU)
コア管理プロセッサ
36
CPUアービタ
40
CPUアービタ
42
コア管理プロセッサ
34
命令メモリ・インタフェース
ハードウェア・クラッシュ・ファイア
52
50
命令メモリ・インタフェース
48
ハードウェア・クラッシュ・ファイア
46
4
チェックサム・コプロセッサ
20
ストリング・コピー・コプロセッサ
22
エンキュー・コプロセッサ
24
データストア・コプロセッサ
26
新値アクセス・ベス・コプロセッサ
28
CAB アービタ
66
CAB アクセス
70
カウンタ・コプロセッサ
30
ポリシー・コプロセッサ
32
カウンタ・マネージャ
72
ポリシー・マネージャ
74
64
出口DS インタフェース
入口DS インタフェース
62
完了ユニット
60

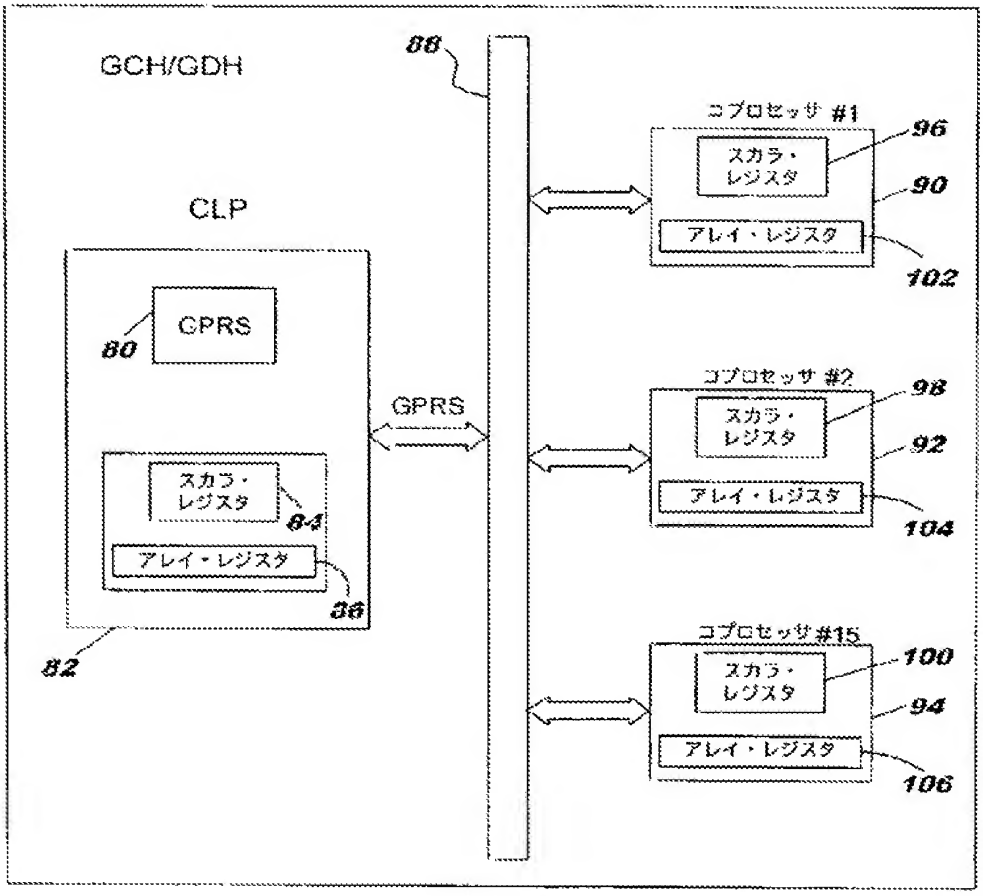
【図2】


10
CLP #1
CLP #2
命令メモリ
ハードウェア・クラシファイア
命令フェッチ/デコード/実行ユニット
汎用レジスタ
専用レジスタ
スカラ・レジスタ
アレイ・レジスタ
ツリー検索コプロセッサ
チェックサム・コプロセッサ
データストア・コプロセッサ
エンキュー・コプロセッサ
CAB コプロセッサ
ストリング・コプロセッサ
ポリシー・コプロセッサ
カウンタ・コプロセッサ
FIFOバッファ
ツリー検索メモリ・アービタ
フレーム・データ・メモリ
フレーム・キュー・メモリ
制御アクセス・バス

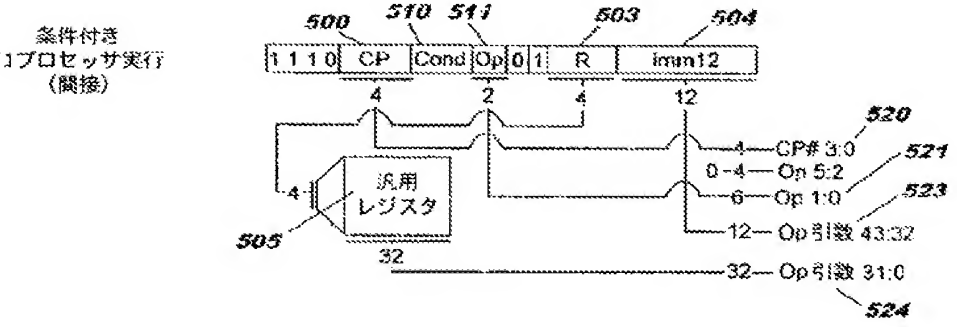
【図7】


コプロセッサ実行（直接）
1111 CP p Op a 0 imm16
CP# 3:0
Op 5:0
Op引数 43:16
Op引数 15:0

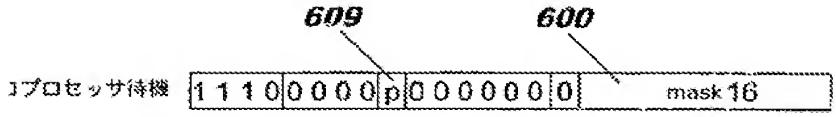
【図３】


GCH/GDH
86
CLP
CPRS
80
GPRS
スカラ・レジスタ
84
アレイ・レジスタ
88
82
コプロセッサ #1
スカラ・レジスタ
96
90
アレイ・レジスタ
102
コプロセッサ #2
スカラ・レジスタ
98
92
アレイ・レジスタ
104
コプロセッサ#15
スカラ・レジスタ
100
94
アレイ・レジスタ
106

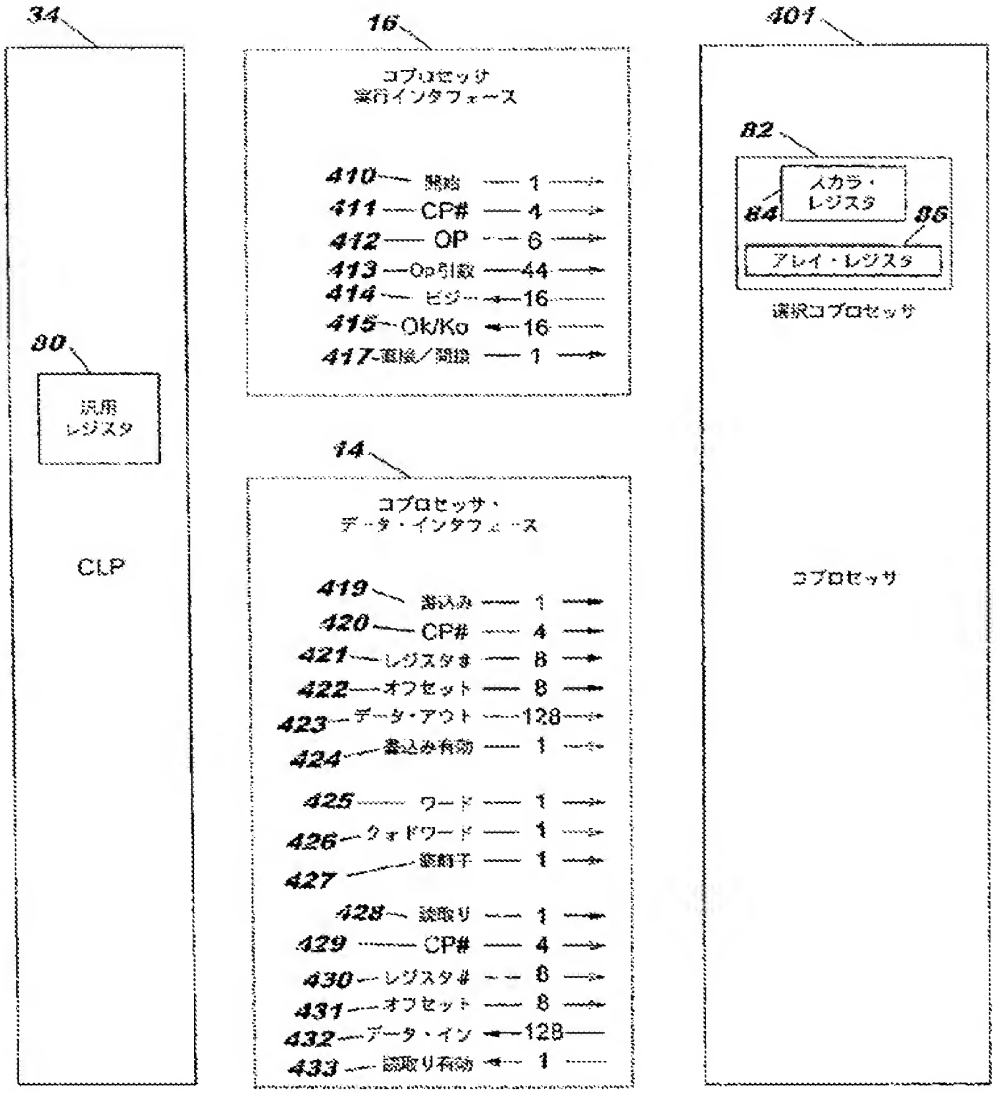
【図６】


条件付き
コプロセッサ実行
(間接)
500
510
511
503
504
1 1 1 0
CP
Cond
Op
0
1
R
imm12
4
2
4
12
520
4—CP# 3:0
521
0-4—Op 5:2
6—Op 1:0
523
12—Op引数 43:32
32—Op引数 31:0
524
汎用レジスタ
4
505
32

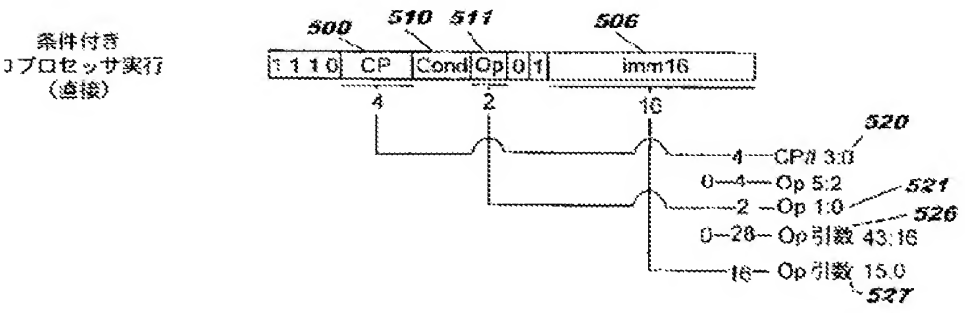
【図９】


609
600
コプロセッサ待機
1 1 1 0 0 0 0 0 p 0 0 0 0 0 0 0
mask 16

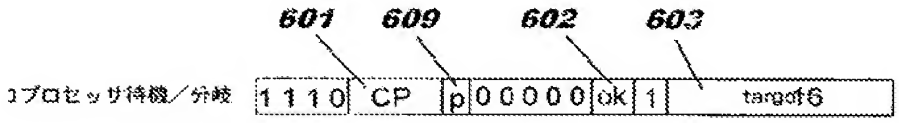
【図4】


34
16
401
コプロセッサ
実行インタフェース
82
410 開始 1
411 CP# 4
412 OP 6
413 Op引数 44
414 ビジー 16
415 Ok/Ko 16
417 要求/開始 1
84
メカラ・
レジスタ
86
アレイ・レジスタ
選択コプロセッサ
80
汎用
レジスタ
CLP
14
コプロセッサ・
データ・インタフェース
419 書込み 1
420 CP# 4
421 レジスタ# 8
422 オフセット 8
423 データ・アウト 128
424 書込み有効 1
425 ワード 1
426 クォドワード 1
427 要求子 1
428 読取り 1
429 CP# 4
430 レジスタ# 8
431 オフセット 8
432 データ・イン 128
433 読取り有効 1
コプロセッサ

【図8】


条件付き
コプロセッサ実行
(直接)
500
510
511
506
1110
CP
Cond
Op
0
1
imm16
4
2
16
520
4 CP# 3:0
0 4 Op 5:2
521
2 Op 1:0
526
0 28 Op引数 43:16
16 Op引数 15:0
527

【図10】



フロントページの続き

(72)発明者 ゴードン・テイラー・デイビス
アメリカ合衆国27514、ノース・カロライナ州チャペル・ヒル、フランクリン・リッジ 97603
(72)発明者 マルコ・シィ・ヘッズ
アメリカ合衆国27612、ノース・カロライナ州ローリー、ナンバー 308、グランド・メナー・コート 4109
(72)発明者 ロス・ボイド・リーベンス
アメリカ合衆国27511、ノース・カロライナ州カーリー、ウィランダー・ドライブ 123
(72)発明者 マーク・アンソニー・リナルディ
アメリカ合衆国27713、ノース・カロライナ州ダーハム、クイーンズバリー・サークル 1201

Fターム(参考) 5B013 DD03
5B098 AA10 GA04 GA05 GA07 GB09 GB14 GC03